100% AMIGA

FREE! Special TECHNICAL Edition Enclosed

Amazing COMPUTING™

Your Original AMIGA® Monthly Resource

For The Commodore AMIGA®

Volume 4 Number 6
US $3.95 Canada $4.95

The *AMIGA* does it all

AMIGA VIDEO PREPARATION

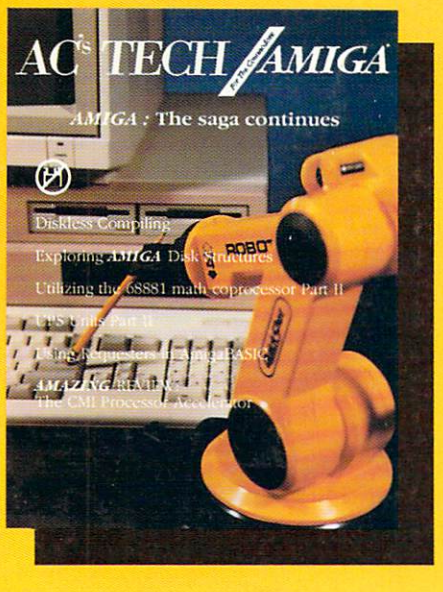
AMAZING Reviews:
KindWords
Nag Plus 3.0

A look at PageStream

ARexx JOINS IT ALL

Plus:

PageStream



Diskless Compiling

Exploring *AMIGA* Disk Structures

Utilizing the 68881 math coprocessor Part II

UPS Units Part II

Using Requesters in AmigaBASIC

AMAZING REVIEW:
The CMI Processor Accelerator

06
74470 74710 9

Fred Fish Collection Reaches 210

A Masterpiece.

Don't limit your potential! Experience **excellence!**, a word processor designed for your Amiga, with 250 available fonts, a Spell-As-You-Type 90,000+ word Dictionary, Grammatical/Style Checker, Thesaurus, Index and Table of Contents generator, Headers, Footers and Footnotes! Sail through PostScript output, True WYSIWYG, automatic Hyphenation, Math, beautiful resizable Color Graphics, flexible Mail Merge, Columns and an easy-to-use Macro-Language making complete actions a breeze! The fastest word processor for your Amiga is the only one you'll ever need! Truly a "Masterpiece" of excellence!

Committed to excellence since 1978

MSS
Micro-Systems Software

12798 Forest Hill Boulevard, Suite 202
West Palm Beach, Florida 33414
407-790-0770 Fax 407-790-1341

See your local dealer for an excellence! brochure.

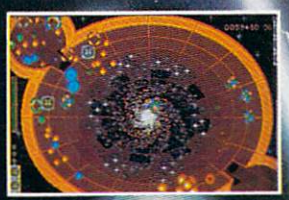
Dealers and Distributors Call 1-800-327-8724

We use KAO Disks!

VORTEX©

BATTLE AT THE EDGE OF THE UNIVERSE

From
The Makers
of **Don Bluth's Dragon's Lair**™
Come TWO Exciting new Games:
Vortex from the author of **C64 ZOOM**™
and
Datastorm from the author of **Sword of Sodan**™.
Now available through your
Local Amiga™ Dealer



The fate of twin universes lies in the balance!

VISIONARY DESIGN TECHNOLOGIES INC.

TO ORDER – CALL OR WRITE TODAY!

Distributor and Dealer enquiries welcomed.

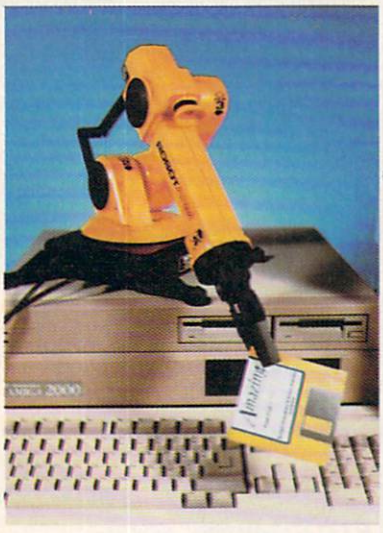
HEAD OFFICE: 45 Whitehorn Cres., North York, Ontario, Canada M2J 3B1 ☎ (416) 497-0833 Fax (416) 497-3077

Product of Canada

Printed in Canada

Amazing COMPUTING™ For The Commodore AMIGA®
Your Original AMIGA® Monthly Resource

AMAZING PROGRAMMING



SVI • 2000A Robot and Amiga Interface
Courtesy of
Datel Computers
3430 E. Tropicana Ave.#67
Las Vegas, NV 89121

AMAZING FEATURES

Your Original AMIGA Monthly Resource

• TABLE OF CONTENTS •

Volume 4, Number 6
June 1989

AMAZING REVIEWS

AMAZING COLUMNS

AMAZING DEPARTMENTS

Amazing COMPUTING™

For The Commodore AMIGA™

ADMINISTRATION

Publisher:	Joyce Hicks
Assistant Publisher:	Robert J. Hicks
Circulation Manager:	Doris Gamble
Asst. Circulation:	Traci Desmarais
Asst. Circulation:	Donna Viveiros
Corporate Trainer:	Virginia Terry Hicks
Traffic Manager:	Robert Gamble
International Coordinator:	Marie A. Raymond
Marketing Manager:	Ernest P. Viveiros Sr.

EDITORIAL

Managing Editor:	Don Hicks
Editor:	Ernest P.Viveiros Jr.
Hardware Editor:	Ernest P. Viveiros Sr.
Submissions Editor:	Elizabeth Fedorzyn
Technical Editor:	J. Michael Morrison
Music & Sound Editor:	Richard Rae
Assistant Editor:	Michael Creeden
Copy Editor:	Aimee Duarte
Copy Editor:	Jan Hammond
Art Director:	William Fries
Photographer:	Paul Michael
Illustrator:	Brian Fox
Production Manager:	Donna M. Garant

ADVERTISING SALES

Advertising Manager:	Alicia Tondreau
Marketing Assistant:	Melissa J. Bernier

1-508-678-4200
FAX 1-508-675-6002

SPECIAL THANKS TO:
Buddy Terrell & Byrd Press
Bob at Riverside Art, Ltd.
Swansea One Hour Photo

Amazing Computing™ (**ISSN 0886-9480**) is published monthly by PiM Publications, Inc., Currant Road, P.O. Box 869, Fall River, MA 02722-0869.

Subscriptions in the U.S., 12 issues for **$24.00**; in Canada & Mexico surface, **$36.00**; foreign surface for **$44.00**.

Second-Class Postage paid at Fall River, MA 02722 and additional mailing offices.

POSTMASTER: Send address changes to PiM Publications Inc., P.O. Box 869, Fall River, MA 02722-0869. Printed in the U.S.A.

First Class or Air Mail rates available upon request. PiM Publications, Inc. maintains the right to refuse any advertising.

Pim Publications Inc. is not obligated to return unsolicited materials. All requested returns must be received with a Self Addressed Stamped Mailer.

Send article submissions in both manuscript and disk format to the Co-Editor. Requests for Author's Guides should be directed to the address listed above.

Amazing Dealers

The following are Amazing Dealers, dedicated to supporting the Commodore-Amiga™. They carry Amazing Computing™, your resource for information on the Amiga™. If you are not an Amazing Dealer, but would like to become one, call PiM Publications, Inc.:

1-508-678-4200

Alaska
That Computer Place — Anchorage
Juneau Electronics — Juneau
Alabama
Abax Data Systems — Huntsville
Alabama Book Store — Tuscaloosa
Madison Book & Computer — Madison
Mel's Photo & Computer Shop — Montgomery
The Computer Nut — Birmingham
The Computer Image — Birmingham
Universal Computer System — Mobile
Arkansas
Commodore Connection — Sherwood
Microtronix — Fort Smith
The Micro Shop — Little Rock
Arizona
Books Etc. — Tempe
Books Brothers Ltd. — Tucson
Computer Library, Inc. — Tempe
Copperstate Business Systems — Phoenix
North American Digital — Tucson
Try Tech Inc. — Pheonix
California
3 E Software And Systems — Hayward
American Mags & Books Ltd. — Englewood
Augnet — Fremont
Balboa Office Supply — San Diego
Bits & Bytes — Fairfield
Bits n Pieces Comp Prod. — Anaheim
Bookland — Marysville
Brown Knows Computing — Redlands
CAL-Tech Bookstore — Pasadena
Candy Computer — Elk Grove
Centerfold News — Los Angeles
Central Park Bookstore — San Mateo
Century Computer Systems — La Habra
Circus of Books — Los Angeles
Circus of Books — Los Angeles
Commodore Land — Alhambra
Commotech — National City
Computer Attic — Palo Alto
Computer Literacy — Santa Clara
Computer Literacy Books — Sunnyvale
Computer Nook, Inc. — San Bernadino
Computer Outlet — Chula Vista
Computer Outlet — San Diego
Computer Outlet — San Marco
Computer Spectrum — Burlingame
Computer Time — Citrus Hts
Cononet Newsstand — Oceanside
Covina Bookstore — Covina
Dataphile — Watsonville
Delaurer's News Agency — Oakland
Discount Computer Station Inc. — Long Beach
Empire Computers — Santa Rosa
Encinitas News — Encinitas
ECX Computer Co. — Walnut Creek
Fisher Computers + Software — Rancho Cucamonga
Harding Way News — Stockton
Heinrichson, Schneider & Young, Inc. — Berkeley
Home Computing Center — Concord
Home Computing Center — Mill Valley
Home Computing Center — San Leandro
HT Electronics — Sunnyvale
International Telecine — Santa Ana
K J Computers — Granada Hills
Little Professor Books — Placentia
LTC Computers — Santa Ana
Lyon Enterprises — Berkeley
Micro Flight — San Jacinto
NewsBeat — Davis
Northridge Newsstand — Northridge
Op Amp Tech Books — Los Angeles
Paras Newsstand — San Diego
Pat's Electronics — North Highlands
R&R Software — Chula Vista
Ridgecrest Computer Ctr. Inc. — Ridgecrest
Sawyer's News Inc. — Santa Rosa
Software First — Santa Rosa
Software Cellar — Anahiem
Software Central — Pasadena
SoftwareLand — San Diego
Software Plus — Citrus Heights
Software Solutions Plus — El Cerito
S.O.S. Computers — Los Angeles
Stacy's #5 — Modesto
Technitronics Service — South Pasadena
The Booksmith — San Francisco
Third Avenue News — Chula Vista
Tower Books — Chico
Tower Books — Citrus Hts.
Tower Books — Concord
Tower Books — Mountain View
Tower Books — Sacramento
Tower Books — San Mateo
Universal News — Hollywood
WCI Computer Central — Eureka
Warner Engineering — San Diego
World Book & News — Hoolywood
Colorado
Aurora Newsstand — Aurora
Citadel Computers — Colorado Springs
Computer Discount — Denver
Computer Repeats — Boulder
Computer Room — Aurora
Micro Computer World — Fort Collins
Micro World — Lakewood
Sun Country Computers — Colorado Springs
Connecticut
Aetna Life Club Store — Hartford
Amiga Business Computer — Norwalk
Computer Outlet — PlaVernon
Computer & Software Connections — Plantsville
Deric Electronics — Hamden
Friendly Computers — Plantsville
Mnemonics Inc. — Norwalk
One Byte — Quaker Hill
Personal Computer Center — Norwich
Software City — Orange
Software Kingdom — East Hartford
Tricom Computers — North Windham
District of Columbia
Programs — Washington
Delaware
Castle Video — Newark
Delaware Electronics — New Castle
Newark Newsstand — Newark
Florida
A A Computers — Jacksonville
Ami Comp Computer Center — Oviedo
Chuck's Newsstand — Vera Beach
City Newsstand — Melbourne
Commodore Country Inc. — Pinellas Park
Computer Bar — Pennsacola
Computer Base — Brandon
Computers Etc. — Sarasota
Computer Image — Miami
Computer Room — Ft. Lauderdale
Computer Terminal Inc. — Sebring
Computers + Video Creations — Palm Beach
Computers 4 Rent — W. Palm Beach
Computers Plus — S. Daytona
Creative Equipment — Miami
Education Computers Etc. — Talahasse
Florida Book Store — Gainesville
Gulf Coast Computer Exchange — PanamaCity
Joe's News, Inc. — Miami
Micro Computer Center — Largo
New Age Electronics — Clearwater
New Age Electronics — St. Petersburg
Newton's Video — Boca Raton
PC Collection — Coral Gabels
Program Store — Clearwater
Prospect News & Books — Ft. Lauderdale
Random Access Computers Inc. — Ft. Walton Beach
RGB Computers and Video Creations — Palm Beach Gardens
Sunny's At Sunrise, Inc. — Sunrise
Computer Room — Ft. Lauderdale
The Open Door — Merrit Island
Georgia
BFJ Enterprises — Roswell
Cardinal Computers — Dalton
Software South Inc. — Savannah
The 64 Store — Atlanta
Hawaii
Astro Computers & Electronics — Aiela
Computronics — Honolulu
Data-One Computer Center — Honolulu
J & C Repair — Aiea
Idaho
A.B.I. Computer & Video — Nampa
Illinois
Audio Visual Req. — Chicago
Comp—Kingdom — Lisle
Computer Software Connection — Moline
DigitalWorld — Addison
DOS Computer Center — Niles
Keep In Pace Computers — Champaign
Kroch's & Brentano's — Chicago
Kroch's & Brentano's — Schaumburg
Kroch's & Brentano's — Orland Park
Kroch's & Brentano's — Auroa
MDJ2 Inc. — Champaign
Micro Age — Oaklawn
Micro Ed Enterprises Inc. — Chicago
Micro Tek — Kankakee
Micro Tech Soultions — Aurora
Ring Software — Geneva
Software City — Bellville
Software Plus — Wheeling
Software Plus Chicago — Chicago
Software Plus West — Hanover Park
The Memory Expansion — Bloomington
Unique Computer — Sterling
Warren Radio/FRANCO — Peoria
Indiana
Bookland — Indianoplis
Burkat Computer Ctr. — South Bend
Computer Corner — Marion
Computer People Inc. — Michigan City
Computer Products Unlimited — Ft. Wayne
CPU Inc. — Indianapolis
Indiana News — Indianapolis
Micro Computer Centers Inc. — Indianapolis
Reese Equipment Co. — Plymouth
The Book Corner — Bloomington
The Computer Room of Merrivile — Merrivile
Vons Computers — West Lafayette
Iowa
Gronert Computers — Des Moine
SGS Search — Ames
Kansas
Arc Electronic Distributors — Wichita
MidKansas Computers — Newton
Midwest Computers — Manhattan
Software City — Overland Park
Kentucky
Computer Dimensions Inc. — Lexington
Computer Mania Inc. — Florence
Hawley Cooke — Louisville
Reality World Computers — Lexington
The Computer Lab — Louisville
Louisiana
City Newsstand — Baton Rouge
Computer Time — Shreveport
Lakeside News — Metairie
Modern Business Machines — Metairie
Software Center International — Metairie
Software Source — Lafayette
The Book Shelf — Slidell
The Computer Clinic — Lafayette
Maine
Computer Barn — Damariscotta
Hands On Computing — Westbrook
Radio Shack Associate Stores — Jay
Skylight Software — Belfast
Maryland
Buried Treasure — Rockville
Cal Com Inc. — Silver Spring
Greetings and Readings — Towson
Knowledgeware Systems — Kensington
Micro Computer Center — Baltimore
Professional Micro Services — Baltimore
Software Advantage — Rockville
Software N Things — College Park
Waldorf Computer — Waldorf
Massachusetts
Computer Zone — North Attleboro
HCS Computer Center — Marshfield
LCA Video & Computer Ctr. — Norwood
Memory Location — Wellsley
OmniTek Computers — Tewksbury
Pioneer Valley Data Equipment — Amherst
Soft Designs — Watertown
Softpro — Burlington
The Bit Bucket — West Newton
The Software Haus — Quincy
Tycom Inc. — Pittsfield
Michigan
CompuSoft — Kalamazoo
Computer Emporium — Lansing
Direct Access — Larsing
Central Park Computers — Okemos
Global Computer Center — Dearborn
I/O Softwware — Flint
Michigan News Agency — Kalamazoo
Michigan Software — Novi
Micro World — Livonia
Micro World — Milford
Micro World — Roseville
Micro World — Southfield
Micro World — Southgate
Micro World — Sterling Heights
Microworld Computer Center — Dearborn
Midland Computer Shop — Midland
Norman Business Control Sys. — Wyoming
Readmore Newscenter — Lansing
Rochester Book Center — Rochester Hills
Roseville Computer Store — Warren
Slip Disk — Madison Heights
Software City — Grand Rapids
Specialists Computer — Livonia
State Street Computers — Ann Arbor
T.J. Enterprises — Clinton
The Computer Programmer — Holland
The Edge Connector — Clio
United Computer — Canton
Ye Olde Computer Shoppe — Ypsilanti
Minnesota
Computer Excercise World — Minneapolis
Computer Stuff — Minneapolis
Computers, etc. — Eagan
JMH Software — Maple Grove
Lookahead Software Compute — rNorthfield
Shinder's — St. Paul
Shinder's — Maplewood
Shinder's — Burnsville
Shinder's — Minnetonka
Shinder's — Edina
Shinder's — Crystal
Shinder's Read More — Minneapolis
Specialists IN — Hopkins
Valiant Inc. — Stillwater
Mississipi
Computer Connection — South Haven
National Computer Center — Ocean Springs
Missouri
Associated Computer Services — Springfield
Brands Mart Computers — Kansas City
Carthage Electronics — Carthadge
Data Byte Computer Co. — St. Louis
Software City — Independence
Systems Plus — Overland Park
Systems Plus of St. Louis — St. Louis
Nebraska
Double E Electronics — Omaha
My Com Inc. — Plattsmouth
Nevada
Computer House — Sparks
Computer World — Las Vegas
Tower Records — Las Vegas
New Hampshire
Dartmouth Bookstore — Hanover
Diversified Computers — Keene
System Eyes — Merrimack
New Jersey
Act One — Vineland
Business System Professional — Scotch Plains
Family Computer Centres — South Orange
Morris County Stationers — Chester
Software Station — Rockaway
The Program Store — Eaton Town
The Program Store — Wayne
Unlimited Software Inc. — Rutherford
New Mexico
New Horizon Computer System — Alamogordo
Page One Newstand — Albuquerque
Road Runner — Espanola
New York
Amuse — New York
Amicus- Hudson Valley — Wappingers Falls
Amiga Business Computer — East North Port
Byte Shop — Merrick
Compute Center — White Plains
Computer Outlet — Jamestown
Computers Etc. — Syracuse
Computerware — East Meadow
Contemporary Computing — Latham
Creative Computers — Johnson City
Cromland Inc. — Brooklyn
Global Software — New Hyde Park
Leader's Video — Schenectady
Leigh's Computers — New York
MCP Associates — Astoria
Mnemonics — North Port
Readers Market — Tonawanda
Software & Such — Scotia
Software City — Forest Hills
Software Link — White Plains
Software Supermarket — Tonawanda
Star Tech Systems — Massena
Synfaire Trading — New York
Sysput Computers — Brooklyn
Tadiran — New York
Tedrow Business Products — Rochester
The Computer Cellar — Albany
Unicorn Electronics — Johnson City
Video Computer Bin — Brooklyn
Video Computer Center — Rome
Village Computer Inc. — New York
World Computers Inc. — Hicksville
North Carolina
Digitz — Raleigh
K&S Newsstand — Clemmons
K&S Newsstand — Kernersville
K&S Newsstand — Winston-Salem
News Center — Cary
Parker News Inc. — High Point
Software City — Raleigh
Software City — Charlotte
TDS Computers — Durham
Triad Computers — Winston-Salem
North Dakota
Computer Associates — Fargo
G. F. Computers — Grand Forks
Software House — Fargo
The Computer Store — Minot
Ohio
BS Software — Columbus
Computer Network — Newark
Computer Site — Cleveland
Eddy's Cigar Store — Elyra
Fairborn Home Computer — Dayton
Lakewood International News — Lakewood
Lakes Consumer Electronics — Akron
Leo's Book & Wine Shop — Toledo
Microtech Computer Center Inc. — Toledo
Microware Magic — Fairfield
News Shop — Kent
North Coast Programming — Willoughby
Paper & Ink — Tiffin
Software Centre International — North Olmsted
Software and More — Cincinnati
Software Connection — Mayfield Hts.
Somers Photo & Comp. — Cleveland Heights
Specialist Computer System — Toledo
Oklahoma
Little Professor Books — Stillwater
Merit Computers — Oaklahoma City
Micro Link Computer Store — Tulsa
P C Tech — Tulsa
Second Hand Software — Oklahoma City
Video-Comp Inc. — Lawton
Oregon
Andmeda — LaGrande
Clackamas Computers — Beaverton
Clackamas Computers — Clackamas
Comm-Shack — Salem
Computron Business Systems — Portland
PC Express — Eugene
Powell's Tech — Portland
Rich Cigar Store — Portland
Sixth & Washington News — Portland
Software Pipline — Beaverton
Tech Books — Portland
Pennsylvania
Alpha Omega EBE — Warren
ATD Software — Sayre
Basic Computer Systems — Erie
Basic Computer Systems — Hermitage
Circle Computer — Ephrata
Computer Wurx — Stroudsburg
Computers and Games — Reading
Downington Computer Center — Downington
Floppy Disk — Mechanicsburg
Floppy House Software — Palmyra
Gene's Books — King of Prussia
Lauer-Todd Computer Solutions — Newtown
Micro Master Inc. — Monaca
North Central Digital Systems — Danville
Pittsburgh Computer Store — Pittsburgh
Software Hut Inc. — Philadelphia
Tasco Computer Center — Lancaster
Triangle Computers — Indiana
Walden Hill Images — Berlin
Rhode Island
Micro Limits — Smithfield
Software Connection — Warwick
South Carolina
Avcom International Inc. — West Columbia
Micro Computer Depot — Sumter
Watson's Jewlery — Chesterfield
Software Haus — Charleston Hts
Tennessee
ComputerLab of Memphis — Memphis
Computer Hardware Services — Nashville
Data International — Hickson
The 64 Store — Bartlett
Texas
All Systems Go Software — San Antonio
Amazing Computer Systems — Carrollton
Bookstop — Houston
Bookstop — San Antonio
Byte Shop — Austin
C.E.M. Corporation — Brazoria
Computer Age — Houston
Computer Magic — Austin
Computer Skills — Eules
FNS Computer Services — Texarkana
Maxwell Book Inc. — Desota
Metropolitan Computer — Richardson
Micro Search — Houston
Software Library — Witchita Falls
Software Terminal — Fort Worth
The Computer Experience — San Antonio
The Computer Stop! — Amarillo
The Federated Group — Mosquite
Utah
Magazines and Gifts — Salt Lake City
People's Computers — Provo
Vermont
Ray Supply Inc. — South Burlington
Virginia
Amy Computer Systems — Virginia Beach
Family Computer Center — Fairfax
Family Video & Computer Center — Newport News
I/O Computer — Norfolk
Software City — Richmond
Tract Software — Prince William
Virginia Micro Systems — Woodbridge
Washington
Amazing Electronics — Spokane
A.P.P.L.E. CO-OP — Renton
Apple Valley Computers — E. Wenatchee
Bits, Bytes and Nibbles — Spokane
Bull Dog News — Seattle
Byte World — Kent
Computer Quest — Edmonds
Computers + Inc. — Tacoma
Family Computers — Everett
Micro Age Computer Store — Bellevue
Nybbles & Bytes — Tacoma
Omni International — Seattle
Port Book and News — Port Angeles
Programs Plus — Seattle
Real Time Systems — Seattle
Take A Byte — Kennewick
The Computer Shoppe — Vancouver
Tower Books — Bellevue
Y Not Magazines and Boks — Federal Way
Wisconsin
20th Century Books — Madison
Computer Software Center — Milwaukee
Fox Valley Personal Computer — Oshkosh
Madison Programmer's Guide — Madison
Mayfair Computers — Milwaukee
Mom's Computers — Stevens Point
River City Software — La Crosse
Software City — Racine
TMW Software Inc. — Wausau
Wyoming
Byte Emporium — Laramie
Computer Works — Laramie
Crazy Mountain Electronics — Casper

Outside of the United States

Canada
Compute or Play — Alberta
Computerworks — Alberta
Computer Shop — Alberta
Data Pro — Alberta
Computer Shop of Calgary — Alberta
The Software House — Alberta
Hindsin Computing — Alberta
Red Deer Micro Systems — Alberta
Memory Land — Alberta
Compu Care — British Columbia
Computer King — British Columbia
Conti — British Columbia
Moebius Computer Systems — British Columbia
Nutec — British Columbia
Pacific Business Technologies — British Columbia
Sarita — British Columbia
Strider — British Columbia
Super Software — British Columbia
Canadian Computer Era — Manitoba
Micromart — Manitoba
ComputerWorld — Manitoba
J.W. Allen — Manitoba
Kobetak — Manitoba
Minicomp — Manitoba
Arkon — Ontario
Chatam Computer — Ontario
Computerland — Ontario
Comspec — Ontario
Electronics 2001 — Ontario
Hastings Data — Ontario
The Computer & You — Ontario
Thornhill — Ontario
Software Co. — Quebec
Deltronix Solare — Quebec
Microdata — Quebec
Info Mereier — Quebec
Laval Micro — Quebec
Vision Tronique — Quebec
Distribution Visipro — Quebec
Distributed in Canada by
Ingram Software Distributors — Concord, Ontario
Compulit Distributors — British Columbia
Phase 4 Dist. Inc. — Calgary, Alberta
Argentina
Logic Computer — Buenos Aires
Australia
Computermart — Leederville,WA
Diskworks — Greenwich, NSW
John Verhoeven — Cloverdale, WA
Maxwell Office Equipment — Abbotsford, Melburn
Run Soft — Melbourne
Saudia Arabia
Abdulla Fouad & Sons — Daman
Scotland
The Amiga Center of Scotland — Edinborough
Sweden
Nicholas Peerson — Lund
W. Germany
Greenville Trading & Import — Oberursel

Amazing Computing™ is also available in most B. Dalton Booksellers, B. Dalton Software Stores, Crown Books, Software Etc., selected WaldenBooks Stores, and Walden's Software Store locations.

From The Managing Editor

A changing of the guard

Most of our long-time readers are aware of the growth of Amazing Computing™. In past issues I have reminisced about the small room and then the basement we started producing AC in over three years ago. It is still hard to believe we have been able to accomplish so much coverage of the Amiga, by beginning from a single desk and phone. I have returned to this early period in our history to acknowledge the efforts of an AC pioneer.

Although Ernest P. Viveiros Jr. did not assist in our first issue's layout, he was there (in our kitchen) when the first shipments of AC were made to our new Amazing Dealers. His first attempts at art and layout began with our second issue. By the third issue, he was a distinctive part of the Amazing staff.

Whether shipping an issue, helping an author, or completing a difficult article, Ernest has always attempted to deliver exactly what is needed. His tasks have ranged from layout design to submissions, and he has handled each with a distinctive brand of common sense and care. For the past year, he has coordinated all our AC authors, as well as the contents for each issue.

Ernest's knowledge of the computing tools available and his constant search for better ways to do our job has made us a more efficient operation. "Don, I think we should buy..." has been both his herald to drive us onward and his humor to maintain our balance. If left alone, Ernest would probably have purchased just about every piece of hardware and software available. His curiosity of new systems and new programs has remained an asset.

Ernest has now decided to serve us all in another capacity. He is joining the U.S. Navy, and this is the last issue he will be putting to press. Although we will miss him, it is easy to see the excitement his new career will offer him. Ernest's actual plans regarding his Navy service and what he will do after basic training are still not determined. One thing is certain however: the Navy will get him up a lot earlier than we ever could.

To Ernest, from all the staff at Amazing, good luck. And from me, thank you for all your help and assistance. You have brought both AC and our coverage of the Amiga into areas we would not have been able to do alone.

Enter the new guard

This month also adds a new Technical Editor to Amazing's masthead. Mr. Mike Morrison, a founder of Micro Momentum, Inc. (the folks who are working on the portable Amiga), has been added to our staff as new Technical Editor to help answer more clearly the many questions we receive from readers each month.

Mike brings his love of the Amiga and his insight into its development to the pages of Amazing Computing. Mike is determined to push the understanding of Amiga principles, as well as the possibilities for the Amiga, to new heights through AC. A lofty and expansive task, still Mr. Morrison began his work at AC saying, "I may not know the answer myself, but I will know where to find it."

Mike's decision to join our expanding group came within hours of the birth of his first child, a ten-pound boy named Alexx. We are not sure whether it was the thrill of the Amiga, or the fear of Alexx's grocery bills that brought him, but we are pleased he is here.

Ms. Elizabeth Fedorzyn has accepted the mantle of submissions editor. She will coordinate any submissions and all author correspondence. This effort will allow us to offer a very close connection with each of our authors as we expand the coverage in both AC and its sister publication, AC's Guide to the Commodore Amiga.

AC's Guide to the Commodore AMIGA?

That's right. Starting in September, AC will publish AC's Guide three times a year—Fall, Winter, and Spring. The Spring Guide will come as an added issue to our regular subscribers. The Fall and Winter Guides will be available as an upgrade to your current subscription or through a reservation plan. (Please see page 73 for more information.)

We have long seen the need for a quality product guide which placed all the available Amiga products in one place, with longer descriptions and cross referencing. As a test, we published the Spring Product Guide to see what effort would be required and how our readers would react. The results were excellent. The Spring Guide has been selling extremely well, and the reaction from our readers is very enthusiastic. With the new schedule of Guides, we will be able to deliver better information to the Amiga community on what is available for the Amiga and how they can best use their machines. We hope this information will not only create a more informed Amiga community, but increase both Amiga computer and third-party sales of hardware and software.

Amazing Computing, volume 4.6

This issue of AC contains a large supplement of technical articles. AC has compiled a wide variety of articles, from AmigaBASIC to Diskless Compiling. When we originally announced our intentions for this issue, we were flooded with responses. We received too many articles to fit in this one issue, so you will be seeing them in future issues of AC.

AC began its career providing as much technical information as it could to readers. In the early days of the Amiga, the Amiga community was comprised mainly of Amiga programmers and computer enthusiasts. As the Amiga grew, so did AC. And while we never stopped printing programs, hardware projects, or technical issues, we continued to increase the size of AC to include more general subjects for the expanding Amiga Community.

AC has maintained this careful balance of general information and technical Amiga information to provide the Amiga public with access to the entire Amiga experience. Each Amiga user has specific ideas regarding what their computer should do. It is in this spirit that we provide an open forum of increasing information. We respect our readership and, thus, try to show them more of the Amiga and entice them to do more.

AC has published more articles, more pages, and more information for the Amiga than any other periodical. We are extremely proud of this heritage and will keep doing what we know best.

Sincerely

Don Hicks
Managing Editor

Got The Picture...Get The Works!

PLATINUM EDITION™

Picture this: all the productivity applications you need in one easy to use "Starter Kit". Give your Amiga power times five. Give it **The Works! Platinum Edition.**

✓**WORD PROCESSOR** • The Works! Platinum Edition word processor is powerpacked with features that help the serious writer excel, in an easy-to-use environment that makes beginners instantly productive. The 104,000+ word dictionary with Scientific and Technical supplements keeps your spelling picture-perfect. The 470,000+ word thesaurus with definitions keeps your word-images precise. Mail Merge eliminates repetitive typing. All this, the ability to print IFF graphics, and more! It may be all the word processor you'll ever need.

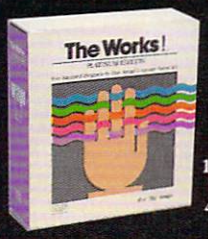
✓**SPREADSHEET** • The Works! Platinum Edition spreadsheet is lightning fast — in fact, the fastest Amiga spreadsheet. And it supports the 68881 co-processor for even more blinding speed. The more than 40 built-in functions do sophisticated calculations. Picture your numbers in any of eight graph types and displayed in eight vibrant colors. Its complete Macro language automates complex operations.

✓**TELECOMMUNICATIONS** • The Works! Platinum Edition is a sophisticated telecommunications program. It is a special part of the outstanding whole. Its multiple terminal emulations and *ten* transfer protocols are the hallmark of flexibility. The script language offers unattended operation and the user defined Macro-Keys reduce complex commands to a single keystroke. Ask about Sadie.

✓**DATABASE** • The Works! Platinum Edition is a flat file manager with power. The extensive mathematical functions make reports much more than a simple list of data.

✓**SIDEWAYS** • The fifth power module in The Works! Platinum Edition, stands your printouts on end. Print ASCII text files and IFF Graphics rotated 90-degrees.

The Works! Platinum Edition is true *integration* from the Micro-Systems Software Development Team; pioneers in Amiga productivity products. Experience full Clipboard compatibility, a common interface, and one user friendly manual. You owe yourself the Platinum experience! Check out **The Works! Platinum Edition** at a dealer near you.

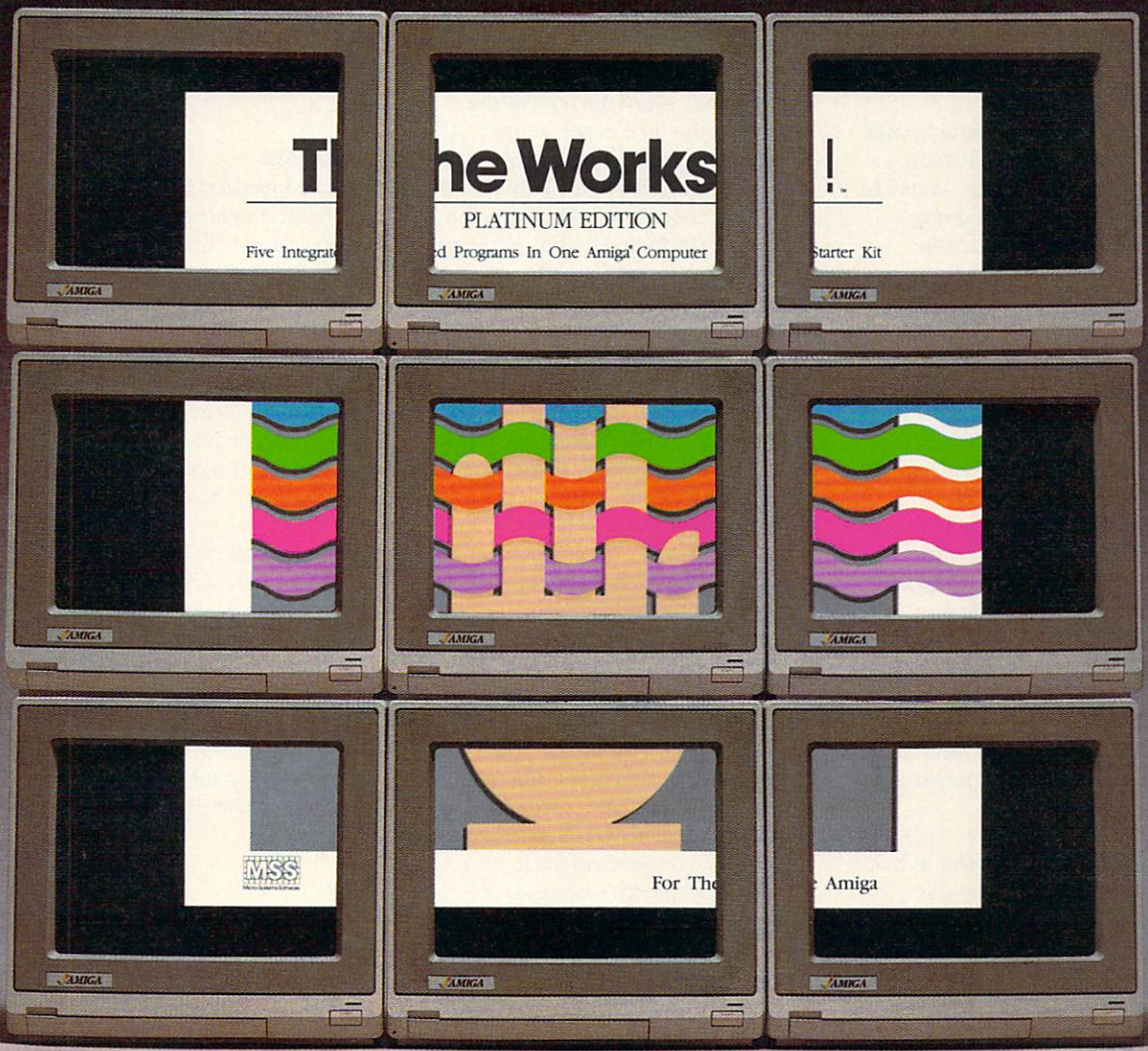


Committed to excellence since 1978

MSS
Micro-Systems Software

12798 Forest Hill Blvd., Suite 202
West Palm Beach, Florida 33414
407-790-0770 Fax 407-790-1341

Dealers and Distributors Call 1-800-327-8724
See your local dealer for a demonstration.
We use KAO Disks.



Amazing Mail

[The opinions expressed in the following letters are not necessarily those of Amazing Computing.AC also reserves the right to edit letters to meet our space requirements.—Ed.]

Dear AC:
Congratulations on a knock-your-mouse-buttons-off *Spring '89 Product Guide*! I will most certainly be turning through the pages again and again. Can you stand the work to make another one?

I would appreciate it if you could mail me a copy of your Writer's Guide. I am interested in writing some articles for you.

Thanks!

Sincerely,
Brian C. Berg
Tempe, Arizona

We're glad you like it! The product guide will become a permanent publication called AC's Guide to the Amiga. *It will be printed in the fall, winter and spring, and paid subscribers will receive the spring issue free. (MM—Tech Ed)*

Dear AC:
The Amiga is (as all of us know) clearly the a superior machine. Any "Amigan" will surely tell you it is the most advanced personal computer ever to come down the pike. So why is it plagued with funky, third rate software? Where is Aldus Pagemaker, Ventura Publisher, and AutoCAD? With its graphics capabilities, the Amiga could handle programs like that better than any other machine on the market. Why is it that Word Perfect Corp. (God bless 'em) is the only "major" software company to develop for the Amiga?

Part of the reason, in my opinion, is that the Amiga is simply not fashionable. It is the wrong computer for its time. It is not a computer for the eighties, but a computer for the sixties! Think about it. The 1960's was a time of great artistic and musical creativity. It was a time of excitement and a time when that which was revolutionary was welcomed and expected. The Amiga is all of these things. It is colorful, musical, exciting, and most of all, revolutionary. In short, it is the perfect computer for a hippie. I suspect that Jay Miner, the guiding spirit of the Amiga, is an old hippie at heart (that's a compliment Jay!) who just happens to be a techno-wiz. Have you noticed, by the way, that Deluxe Paint is written by Dan Silva, who wears little wire-rim glasses and lives in the San Fransisco Bay Area. Hmmmmmm... If, through some twist of time, the computer revolution had happened in, say the 1961 instead of 1981, things might have turned out quite differently for the Amiga.

But this, alas, is not the time of the hippie but the time of the yuppie. It is a time of mediocrity, not creativity. It is a time when the gray-suit is king and can dictate his tastes to us. He likes money, dull music (including dull rock and roll), more money, uninspired art, spreadsheets, mediocre politicians, still more money, and he doesn't like Amigas. He doesn't like them because they really are best suited for creating art and music and he sees them as frivolous...a waste of time... can't make money at it. He doesn't particularly like the people who like Amigas either. They are artists, musicians, and (yuk!) programmers too. He thinks there is something "squirrelly" about them...a bunch of flakes and crackpots. Deep down, though, he doesn't like them because they are creative while he is dull and unimaginative. They make him feel inferior and he likes that least of all.

So here we have the gray-suits at Commodore who, by some fluke, got hold of the Amiga. It is a machine they don't like or understand made by a bunch of people they don't like or understand (who they fired), that they have to try and sell to another bunch of people who they don't like or understand. How do we know they don't like it? Because they keep trying to turn it into a Frankenstein's Monster full of IBM clone stuff. Can you picture Apple doing that with the MacIntosh? If I were an armchair shrink I might even go as far as to say that they resent and misunderstand the Amiga so much that they don't even like to advertise it. They are ashamed of it and wish it was an IBM! After all, an IBM is a much more "respectable" machine.

But they are stuck with it (and we with them), and we can only hope that they can someday learn to love the Amiga as we do (yeah, right!). Until that time, all I can say is Peace! Love! and happiness....

Gregory LeVasseur
San Fransisco, CA

Dear AC:
While I realize your *Amazing Mail* column is not normally used for publicizing bulletin boards, I feel this case warrants an exception.
For whatever reason, free, privately owned BBSs tend to come and go very quickly, most likely due to a combination of the hard work and the cost involved in keeping them online. I could name only a dozen non-pay boards that I feel are both run professionally and represent a permanent installation.

Many boards pick a familiar theme and customize their menus and formats appropriately. I won't go into the lengthy details that would be needed to explain fully my recommendation of "The Amiga Sharewarer's BBS". The sysop, Bill Beogelein, is trying to make a single location the headquarters for contacting shareware authors about their software, as well as making the very

(continued)

Lattice Tools & Libraries

We are the company that writes the language and the tools. So who could know more about the utilities you need to maximize productivity?

Lattice 5.0 Cross Compiler
All the power and facilities of the Lattice AmigaDOS 5.0 compiler in a cross-compiler MS-DOS and OS/2 host. Includes our full screen cross debugger. $750.

Lattice C++
Object Oriented C++ Programming for the Amiga. Provides object definitions for AmigaDOS, EXEC and Intuition. $300.

Lattice Compiler Companion
Collection of UNIX-like utilities that make your programming environment more productive. (Included in Lattice 5.0.) $100.

dBC III Library
Amiga programming library that provides programming interface into dBASE III compatible files. $150.

Communication Library
New Amiga programming library that supports modem interfaces for XMODEM, YMODEM, KERMIT and ASCII protocols. $250.

PANEL Library
Amiga programming library that supports windows graphics applications. $195.

C Programming Seminar
Learn C programming from Lattice experts. Four day seminar in Chicago. $895.

Call any of our world-wide distributors, or call us at 1-800-444-4309 or (312) 916-1600. Fax #(312) 916-1190, TELEX 532253.

"Brilliant!"
– *William Hawes* author of ARexx

"Lightning Fast!"
– *Reichart Von Wolfsheild* Silent Software, Inc.

"Blows away all the others."
– *Robert Berryhill* LMR Creations

Lattice AmigaDOS C 5.0 is getting lots of praise. And not just because of the latest benchmarks which prove Lattice C 5.0 is the best compiler by every measure.

For instance, Michael Berenstein calls 5.0 "Wonderful," while Roger Uzun simply calls this total programming environment "...the best."

Tom Dolan felt the same about CodePRobe, our new Source Level Debugger. Support for both C and Assembly language, plus multi-tasking debugging make it "...the best I have seen or used!!!!"

And our new Global Optimizer caused Steve Tibbett, a "VERY happy customer," to say, "WOW. This is outstanding!" and Tomas Rokicki of Radical Eye Software to exclaim "...Global Optimizer, inline subroutines and extensive libraries make whole new levels of performance possible."

About our manual? "Excellent!" And "...extra fine."

About our BBS for unlimited technical support? "Fantastic!"

And about our competition? Eric Dyke of Inom Software says, "Manx, eat your heart out! Lattice is the right choice!"

Lattice
Professional Programming Tools Since 1981
2500 S. Highland Ave., Lombard, IL 60148

latest versions of their work available under one roof. It is a format that is long overdue. Nice job, Bill.

Give them a call at 313-473-2020. If you use PC Pursuit it's a local call via node "MIDET".

Sincerely,
Pete Raddatz
Detroit, MI

It's people like Bill who help keep the Amiga community filled with easily available information. (MM— Tech Ed)

Dear AC:
Here is a small tip that I hope you will be able to put in your magazine.

On the Amiga 500 the joystick/mouse ports are at the rear and impossible to see to plug in joysticks, mouses or dongles.

A cheap fix for this is to purchase a joystick port extender (Radio Shack and other electronic stores sell them). Simply leave the extension plugged in and route the cord to an easy spot to reach on your desk and when you have to plug something in, all you have to do is plug it into the extension, instead of fumbling around the back of the computer; thus eliminating damage to the ports or connectors.

Yours Sincerely,
Stuart Attwood
Canada

Dear AC:
In reference to *Amazing Computing* Vol 4 No. 2 Crunchy Frog. A nice article, but one error in the listing on page 102, right hand column, about 15 lines down:

struct Window *window_ptr = ...

The asterisk was omitted from your listing.

Also, in reference to Amazing Letters, the ELSE bug, page 8. This is more exactly an ELSE .. END IF bug, and isn't hard to correct. It results from accidently leaving spaces behind the IF of the END IF; such spaces should not bother AmigaBASIC, but they do.

Simple program example:

```
FOR j=1 TO 10
  IF j MOD 2 = 0 THEN
    PRINT j;" is even."
  END IF      (see note below)
NEXT j
```

Deliberately leave a few spaces at the end of the END IF line. No problem when you run. Next, just before the END IF line, add:

```
ELSE
  PRINT j;" is odd."
```

Oops: the computer beeps and comes up with the error requestor. Note that the orange rectangle surrounds the extra spaces you typed.

Click on OK and delete the spaces, using either the backspace or the delete key. Now your program will run.

It's annoying, but not crippling. The ELSE statement is too useful to abandon just because of this irksome anomaly.

Jim Butterfield
Toronto, Canada

We recieved several letters pointing out the error in the Crunchy Frog program. Thanks. Also, thank you for helping to clear up the ELSE .. END IF bug in AmigaBASIC. (MM— Tech Ed)

To Whom it may concern:
Although I don't own an Amiga, I have found your magazine to be one of the most thorough of all the Amiga magazines on the shelf. As an Atari ST owner, I haven't always found the Bandito's comments to be especially useful, but overall, I think his/her column is a definite plus for your magazine.

Keep up the good work,
Gordon Meyer

Dear AC:
I wish to relate a pleasant experience that I have received from one of your advertisers.

I purchased an Expansion Technologies, Escort 32 Meg Hard Drive for my A1000 which attaches to the Buss that already had 2Megs connected. The only alteration that I had to have done was the PAL ground fix.

Since its intallation with WB 1.3 and FFS, my system is flying. The technical assistance from Expansion Technologies is excellent and the quality of the product is top notch...

Sincerely,
Bruce Donally
Brighton, MA.

Dear AC:
I would first like to say that you guys do a fine job. I hope you keep up the good work and continue publishing this valuable resource. I have yet to receive an issue of *Amazing Computing* that didn't keep me busy trying out the new hints, programs and tricks. By the way, I like your new title.

I would now like to announce my BBS so you can put it in your BBS list that you publish in your next product guide (I can't wait!)

The Guru BBS
(414) 582-7448
9600 baud

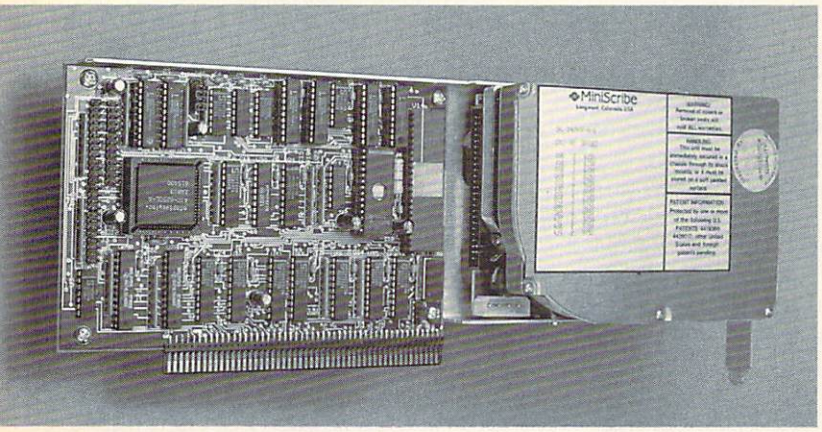
I am not hoping for any intercontinental callers but I just want Wisconsin Amiga users to know that there is one more Amiga BBS that they can add to their short list.

Sincerely,
Erik Meitner
Winneconne, WI

(continued)

HardFrame/2000

The *Super-Speed*, DMA, SCSI Hard Disk Interface for the Amiga®2000



- **AutoBoots AmigaDOS 1.3 *(Price Includes HardFrame Eprom!)***
- ***Directly* Boots the New Fast-File System! *(Doesn't Need Old FS!)***
- **Auto-mounts All Hard Disk Partitions *(no Mount List Required!)***
- **Designed-in, Ultra Strong, Multitasking Performance**
- **High Quality Metal Frame for Stable, On-Card, Hard Disk Mounting**
- **Power Cabling Directly from Card to Disk**
- **50-pin Cable Included**
- **Supports up to *seven* SCSI hard disks of any size**

How fast is *fast*? **HardFrame/2000** transfers data at Amiga bus speeds! It's actually faster than the hard disk mechanism itself! And even more important in the Amiga's multitasking environment, **HardFrame/2000** has extremely efficient DMA circuitry to get on and off the bus in almost no time at all: 280ns to get on; 200ns to get off. And it's true, dedicated DMA, too! **HardFrame/2000** autoboots *and* automounts *directly* into the AmigaDOS™ 1.3 Fast File System (old file system partitions are *not* needed!). The core of any DMA SCSI interface is in its SCSI protocol chip and DMA chip. MicroBotics has chosen the new, high performance Adaptec AIC-6250 SCSI chip, capable of up to 5 megabytes per second raw transfer speed, and the Signetics 68430 DMA chip running at 12.5 megahertz. Then we added additional FIFO buffering and enabled 16-bit wide data transfers for maximum throughput. The sophisticated design of **HardFrame/2000** provides for automatic SCSI arbitration, selection and reselection. The hardware supports either synchronous or asynchronous data transfer. **HardFrame/2000** can function as either the SCSI bus initiator or the target and can reside in a multiple master environment. Physically, **HardFrame/2000** is optimally flexible: the compact, half-size card comes attached to a full length, plated aluminum frame. The frame has mounting holes positioned to accept standard, 3.5" SCSI hard disk units such as those manufactured by MiniScribe, Seagate, Rodime, and others (hard disk mechanisms must be supplied by the user or his dealer as a separate purchase item). Alternatively, you can cable-connect to a SCSI drive mounted in your Amiga's disk bay or in an external chassis. As many as seven hard disks may be connected to a single **HardFrame/2000**. There is no size limit on each disk. **HardFrame/2000** includes a 50-pin SCSI cable and header connectors for either 50-pin or 25-pin cable connection. Also included is a current tap to power frame-mounted drives directly from the slot itself. **HardFrame/2000** comes complete with driver, installation, and diagnostic software.

Available NOW! Suggested list price, $329 (hard disk not included)
Frameless version: $299.00. See your Amiga Dealer.

The HardFrame/2000 photo shows the product with a MiniScribe twenty megabyte hard disk installed. Hard disks are *not* included in the purchase price of HardFrame. Note that if placed in the first slot, HardFrame uses only one slot even with a disk attached.

New!
8-UP! (DIP) FastRAM

Another great memory board from MicroBotics, 8-UP! (DIP) is the "brother" of the original 8-UP! (which uses SIMMs and PopSIMMs to fill its memory space). 8-UP! (DIP) uses conventional 1 megabit RAM chips in standard sockets to provide your Amiga 2000 with 2, 4, 6, or 8 megabytes of autoconfiguring FastRAM! 8-UP! (DIP) is a super efficient CMOS design for lowpower consumption and high reliability. Suggested list price, $239 (0k installed)

Join MicroBotics
ONLINE TECHNICAL SUPPORT
CONFERENCE ON BIX
(The Byte Information Exchange)
–call 1-800-227-2983
for BIX membership information!

MicroBotics, Inc.
Great Products Since the Amiga Was Born!
811 Alpha Drive, Suite 335, Richardson, Texas 75081 (214) 437-5330

Tell your dealer he can quick-order from MicroBotics directly – no minimum quantity –show him this ad!

Dear Sirs:
Ever since I purchased my first computer, a PET, in 1989 I have been a dedicated Commodore user. At the moment I operate INFOMATIQUE! an Amiga based BBS. To the best of my knowledge INFOMATIQUE! was, with the exception of Commodore's developers BBS, the first Amiga based BBS in Europe. It was certainly the first in the world to use Sidecar (I had Sidecar a long time before it reached Nort America).

Right from the start I have used BBS-PC! and I have been very happy with it but I am now beginning to feel a bit restricted by this software and I am, therefore, interested in finding new software. I have a particular interest in setting up a multi-user system.....is there any suitable software/hardware available or in the pipeline. All information from readers and software developers would be appreciated.

There are only a few bulletin boards in this country and as a sysop I feel somewhat isolated so I would really like to make contact with other sysops throughout the world so that we may exchange ideas etc.

Your publication is one of the best available.....keep up the good work.

Regards
Liam Murphy
IRELAND
FAX 423123
PHONE 423055
BULLETIN BOARD 764942

Amiga Users:
I have been an Amiga owner for the last year and a half. I originally upgraded from an 8-bit Atari 130XE.

After purchasing the Amiga, I was thrilled to hear that Word Perfect gave support to the Amiga not with ONE program but TWO. Those programs are Word Perfect 4.1 and Library. I was thrilled with pride, and my heart leaped, that Word Perfect Corp. decided to support the Amiga, it gave me a sense of security knowing that an IBM giant would support the Amiga. Word Perfect gives Amiga a step into the business market, it was the first major company to come in and support the Amiga.

Hopefully other corporations will follow suit like: Microsoft, Borland, and Ashton Tate.

Word Perfect gives the Amiga some of the recognition it whole heartedly deserves. With the continued sales of both Word Perfect and Library, a customer representative from Word Perfect told me Plan Perfect and Word Perfect 6.0 would be the future planned releases on the Amiga platform! That is not a misprint, that is not a misprint, six point oh, on the Amiga before the IBM version!

Again my heart leaped for joy, Amiga would be even further recognized as a more "serious" computer with the coming event of Word Perfect 6.0. The Amiga will be the developing platform for six point oh. With true "what you see is what you get" (wysiwyg), even with five point oh on the IBM you would still need to use page preview or a Hercules Graphics Card with Ram Font.

The most important aspect was the fact that Word Perfect would have had six point oh on the Amiga first. This sadly, is no longer the case. According to another representative who I spoke with today, Word Perfect Corp. has decided to "temporarily" halt Research and Development for Plan Perfect and Word Perfect 6.0. According to him this is due to the slow down in sales of Word Perfect and Library.

Amiga Users far and wide, my point is not to tell you to immediately go out and purchase Word Perfect, rather my point is that we should rally together and send in letters to Word Perfect as a campaign to show that there are many users out here who would be willing to PURCHASE Word Perfect 6.0 for the Amiga as soon as it is available.
I also suspect that another reason for the recent halt in "development" is due to piracy. Piracy is a part of all computers form 8 bits to 16 bits, that is one of the main reason why the support for "OTHER" computer versions has died.

As a new Amiga owner I feared that Amiga was following in the same path, but thankfully for the release of Library, my fears were allayed. Now, my fears are back. As I have said, Word Perfect representatives at Ami Expo in New York and also in various computer shows confirmed that there would be Word Perfect 6.0 and Plan Perfect out for the Amiga. If we could all do a 'write in compaign', I'm sure that Word Perfect would take notice. It doesn't have to be anything elaborate. Just a simple note stating that you are an Amiga user and would be interrested in purchasing Word Perfect 6.0 when it becomes available.

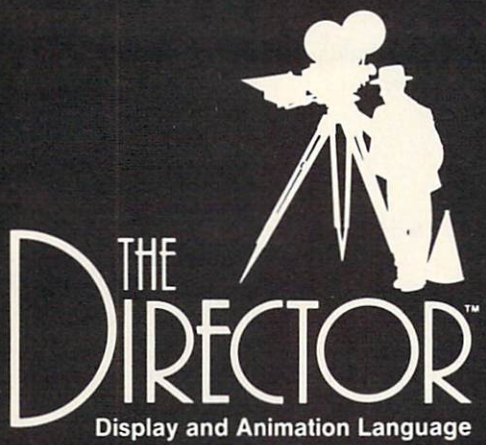
We are now 1 million strong and with one million letters being sent into Word Perfect's mail box, support will resume!

Alex Yang
Whitestone, NY.

Nothing would lend more credibilty to the Amiga as a business machine than having Word Perfect 6.0 and Plan developed and available on the Amiga FIRST! Let's hope that Word Perfect Corporation will continue developement of these two programs on the Amiga. (MM— Tech Ed.)

•AC•

Compatible with Dpaint III™ animation . . .



"If an Oscar were to be presented for Technical Excellence in Amiga Graphics, the winner would certainly be (the envelope, please) - The Director . . an exciting, unique program . . likely to become a classic . ."

Steve King,
Commodore Magazine
April 1988

"I must give The Director top marks for ease of use and capability. For the novice or serious presentation creator, this package is unequaled. It belongs on the shelf of anyone who considers himself an Amiga graphics connoisseur."

Oran J. Sands III,
Info Magazine
June 1988

*. . . **And that is only the beginning.** In addition to giving you frame by frame control over multiple Anims, **The Director** can add page flipping, sound, text generation. and mouse or keyboard interactivity to your presentations. Create anything from the simplest slideshow to the most sophisticated desktop video production.*

*Script your production with an easy, Basic-like vocabulary. **The Director** provides powerful image and memory management, blitter, text and effects control. A freely distributable player program, the Projector, is also included.*

- Use IFF images in any standard resolution including HAM and overscan
- Preload images, Anims, fonts and sounds up to your memory limit
- Basic-like vocabulary: For/Next, Gosub/Return, If/Else/Endif
- Arithmetic expressions, random number generator, variables
- Execute AmigaDOS commands from your scripts
- Fades, Dissolves, Blits, Wipes, Stencils
- Page flip full or partial screens
- Text string and file input and output
- Keyboard and mouse interaction
- Drawing and palette commands
- Digitized soundtrack module
- Supports IFF Anim playback
- PAL compatible
- Not copy protected

$69.95

DEMO DISKS
$10.00 each
Probe Sequence (512K)
RGB (1 meg)

NEW! DIRECTOR TUTORIAL VIDEO $39.95

A step by step guide to using The Director. The tape takes the novice through AmigaDOS CLI commands, script editing, adding effects to slideshows, and page flipping animation. The more experienced user will learn double buffering, effects with Anims, the sound module, the array, and advanced techniques.

TOOLKIT for THE DIRECTOR $39.95

The Director Toolkit is a disk packed with features and enhancements to expand the capability of The Director. There are new wipe routines, a palette selector, a pie chart generator and much more. The new and enhanced BLIT Utility has a powerful interface to help create Wipe, Dissolve, and BLIT operations. It also automates the process of moving an object over a background, generating a complete working script.

This disk is intended to be used with The Director software.

- New wipe routines
- Enhanced BLIT Utility including object movement over backgrounds
- Standard file requester callable from Director scripts
- Screen save from Director scripts
- MIDI input module
- Standard Anim compressor
- Pie chart generator
- Sine and cosine functions
- Card game example
- Palette selector
- Text displayer
- And more!

Check or money order payable to:

Right Answers
Box 3699
Torrance, CA 90510
(213) 325-1311

Please add $3 shipping and handling
California residents add 6½% sales tax.

New Products and

Gold Disk Software's **Design 3D** brings the power of professional animation to your Amiga. Design 3D is an interactive 3D object editor with rendering and animation capabilities. The program lets you create three-dimensional animation ojects which you can add up to four light sources you define, and you can manipulate your point of view at the click of a mouse.

The screen is divided into four work windows, each displaying a particular view of your object: front, side, top, and perspective, which features hidden surface removal and automatic shading from any viewpoint. Design objects by pointing and clicking the mouse, and by using the tools at the top of the screen.

Pull-down menus allow different configurations for your designs, and they offer tools to make your designing easier. Cursor coordinates are displayed continuously at the bottom of the screen in real time for precise object construction. You can save and load objects in VideoScape format, which means you will have a powerful 3D object creation tool to use with Videoscape.

Design 3D lets you create wire frames and solid objects in color, auto 3D text with built-in font editor, built-in animation facilities, and real-time object rotation. The program reads and writes VideoScape objects, creates files for Professional Page, lets you use genocks, or record animations on a VCR.

When you are through, you can output to printers or an HPGL plotter. Whether you are an artist, designer, architect, engineer or home user, Design 3D will add another dimension to the objects you render.

Prison

The year is 3033, and crime is so rampant that criminals are no longer being sent up the river. Instead, they're being sent up in space, to the abandoned planet Altrax. And you, lucky soul that you are, get a free ride on the Altrax shuttle when you're unjustly accused of a crime.

That's when the fun begins. In Actionware's **Prison**, you're job is to escape the odious place before they ship you back in a shuttle coffin.

Fortunately for you—but unfortunately for the driver—a pleasure craft has crash landed on the planet's electro-security net, wrecking the vehicle but leaving the escape pod intact.

There's one problem though. You don't know where the pod is. And of course, a few other people—namely every prisoner on the planet—want the craft, too. So you must kick, punch, and claw your way through maore than 300 screens of play as you search for the pod.

All movement is controlled by the joystick. There is no typing involved. Your character can communicate and interact with other characters—usually by a swift kick to the groin.

Uzzi Interface

If you need help getting off the planet, grab an Uzzi. With Micro Momentum's **Uzzi Interface**, you can do just that. The Uzzi Interface plugs between your computer and joystick, and allows you to switch between standard and rapid-fire mode.

In single fire mode everything works normally. But in rapid fire mode, with the fire button held down, a continuous stream of bullets (or whatever type of ammunition your game uses) will rain down on your enemy. You can use the fire rate frequency knob to fine tune the fire rate from 1 to 20 times/second, depending on the game you are playing.

The Uzzi Interface comes with a standard 4-ft. cable to accomodate Amiga 500 owners. It is fully compatible with the standard mode, or your favorite joystick.

Serial Solution

The **Serial Solution,** from Checkpoint Technologies, is a dual port serial board for the Amiga 2000. The Serial Solution is an internal plug-in board that adds two serial ports to the Amiga's built-in serial port.

The first port is an Amiga-compatible, 25-pin serial port that will drive most serial devices. The 24-pin port supplies 12 Volts of power, just like the Amiga's built-in port, so it will accomodate most Amiga-specific peripherals.

The second port is an AT-compatible, 9-pin serial port. The 9-pin port is functionally equivalent to the 25-pin port, but it will also adapt to AT serial cables. The Serial Solution can be used with printers, plotters, laser printers, PostScript printers, modems, MIDI interfaces, drawing pads, sound samplers, and VCR controllers.

The Serial Solution works with most software written for the Amiga's internal port. With a port configuration program, you can use your software with the 25-pin or 9-pin port, so you can print, plot, play MIDI instruments, and be on-line at the same time.

The board plugs into any available expansion slot on the Amiga 2000. The Amiga's auto-config services will map the board into the Amiga's device address space.

Break into higher storage capacity

Progressive Peripherals' **The Vault** gives your Amiga 500 or 1000 a standard interface to use standard low-cost, IBM-style hard drives. The Vault uses RLL technology to store more information on each track of your hard drive, giving you higher storage capacity at a lower cost.

Other Neat Stuff

A variety of utility software will help you format and maintain your hard drive. The Utility Manager simplifies installation even for the novice. The Vault also includes the CLImate utility program to help you manage the contents of your hard drive.

The Vault's interface cable adjusts to the difference between the Amiga 500 and 1000 expansion buses so you can connect The Vault to either a 500 or 1000. The cable also lets you place the Vault up to 8 feet from the computer.

The package comes with The Vault hard-drive cabinet with hard drive, The Vault Intelligent interface cable, installation manual, installation software, power cable, CLImate disk management software. The Vault comes in four sizes: 20 meg ($599.95), 30 meg ($729.95), 40 meg ($849.95), and 65 meg ($1099.95).

Designer Databases

For the hacker who has everything, Software Visions proudly presents **Designer Databases** for home and business use. The ready-to-use databases are sold separately. They work with Microfiche Filer and Microfiche Filer Plus, and provide examples and solutions for common home and business problems.

The Business I database contains the first commercially available fourth-party ARexx macros. The Inventory form provides automatic processing and reporting macros; billing features automatic inventory and updating macros; and the calendar has automatic reminder and reporting macros. The mail merge feature provides macros for Word Perfect, ProWrite, excellence!, scribble!, and Kindwords.

The Home I database provides forms to keep track of your videotapes, CD's, records, stamps, coins, books, recipes, wines, personal inventory, home budget, and Fred Fish Disks.

Software Visions plans to expand the collection soon with Video/Graphics/Sound I, Home II, and Business II.

Momentum Check

Also on the financial management track is Micro Momentum's **Momentum Check**, a personal checkbook management program. Momentum Check's transactions are limited only to disk space, and can print checks with custom setup option, provide full-screen editing on forms, and support any Amiga hardware configuration.

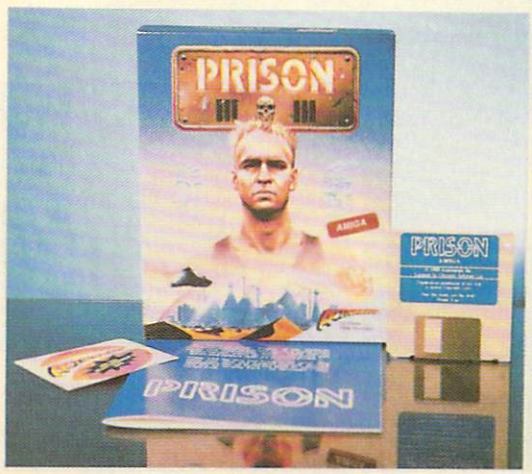
A utility menu of functions will let you change posted transactions in the Master Ledger and provides a backup restore function.

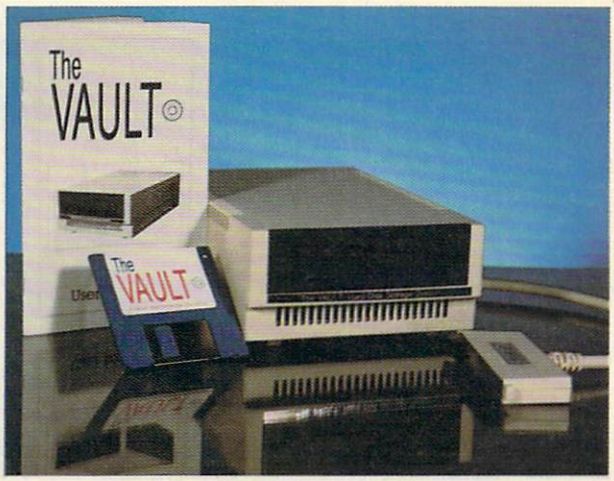
You can inquire and report functions by date, check number, and classification code. Class codes are set up by the user to track different money transactions.

For example, you could set up a Class Code for entertainment that would allow you to generate a custom report of all the money you have spent on entertainment by date or check numbers. You can also set budgets for each Class Code and then generate reports showing actual verses budget. The Addendum feature lets you track cash transactions that may affect your budget or reports. If you pay for a meal with cash, you could use an Addendum so that meal would be included in your entertainment report.

Momentum Check also provides identifiable screens for different functions. When you do a deposit, you get a deposit slip; a withdrawal provides a withdrawal slip, etc.

Balancing your checkbook is a snap because Momentum Check prompts you for all transactions that appear on your bank statement. Once that information is entered, your checkbook is balanced.





(left)
Prison from Actionware

(right)
The Vault from Progressive Peripherals

Momentum Check provides you with an easy-to use-programs that can help you keep your checkbook and budget in line.

Hardrive from Melmac

For those of you who think A.L.F. is just a furry freak from outer space, Pre'spect Technics presents **A.L.F. (Amiga Loads Faster).** A.L.F. is an adaptor kit that lets you easily connect low-cost, IBM-standard hard disks such as the ST-412/ST 506 to the Amiga. The kit includes both the hardware and software you need to get your hard disk up and running in minutes. The hardware unit includes a standard IBM hard disk controller and an adaptor to connect the controller through the Amiga. A.L.F. also lets you externally connect two hard disks to the Amiga.

A.L.F. features shorter loading times, automatic installation, increased writing rate and higher data safety. A.L.F. also includes AlfMount, an Automatic Mountlist editor, and many additional utility programs. The hard disk is also write-protectable.

You don't have to know complex CLI commands to mount the hard drive, as all utility programs can be run from the Workbench.

Version 2.0 should be available soon, and will include reset-protected driver with AutoMount. No mountlist and no change of startup-sequence will be required.

Product Information

Gold Disk
P.O. Box 789
Streetsville, Mississauga, Ontario
Canada L5M 2C2
Tel: (416) 828-0913
Design 3D, $ 99.95

Actionware, Inc.
38 W 255 Deerpath Road
Batavia, IL 60510
Prison, $ 39.95

Micro Momentum, Inc
100 Brown Avenue
Johnston, RI 02919
Uzzi Interface, $34.95
Momentum Check, $29.95

Checkpoint Technologies
P.O. Box 2035
Manassas, VA 22110
Tel: (703) 330-5353
The Serial Solution, $299.00

Software Visions, Inc.
P.O.Box 3319
Framingham, MA 01701
(508) 875-1238
Designer Databases
Home, $39.00; Business, $59.00

Progressive Peripherals and Software
464 Kalamath Street
Denver, CO 80204
Tel: (303) 825-4144
The Vault

Pre'spect Technics Inc.
P.O. Box 679
Station 'H'
Montreal, Quebec H3G 2M6
A.L.F.

Microdeal
576 S. Telegraph
Pontiac, MI 48053
(313) 334-8729
Airball, $39.95

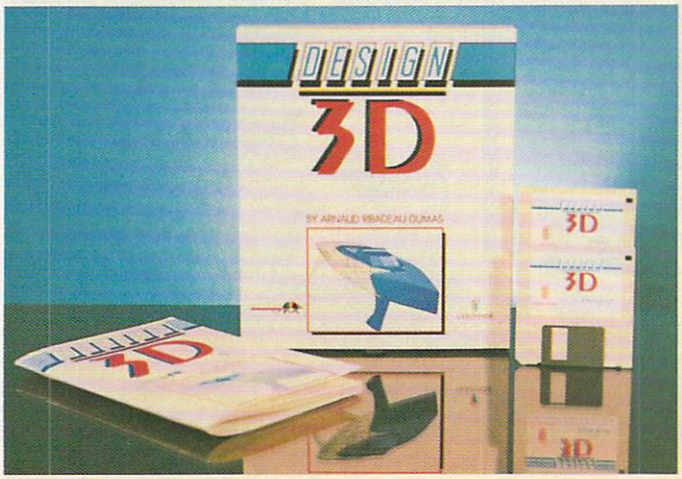
Airball

Another basketball simulation? Well, sort of. In Microdeal's **Airball,** *you* are the basketball simulation. The Evil Wizard has turned you into a rubber ball with a slow air leak, then set you loose in a mansion with 300 rooms. Your goal: find the spell book that will set you free—and don't stray too far from the air pumps.

Somewhere in the castle is a spellbook with the incantation that will bounce you back into human form. And you will have to keep an eye out for the air pumps that will provide you with precious oxygen.

To keep you on your toes, the rooms are filled with spikes of all types and sticky spots on the floor that'll tear you up—literally. As you bounce along, you will pick up objects needed to add points to your score.

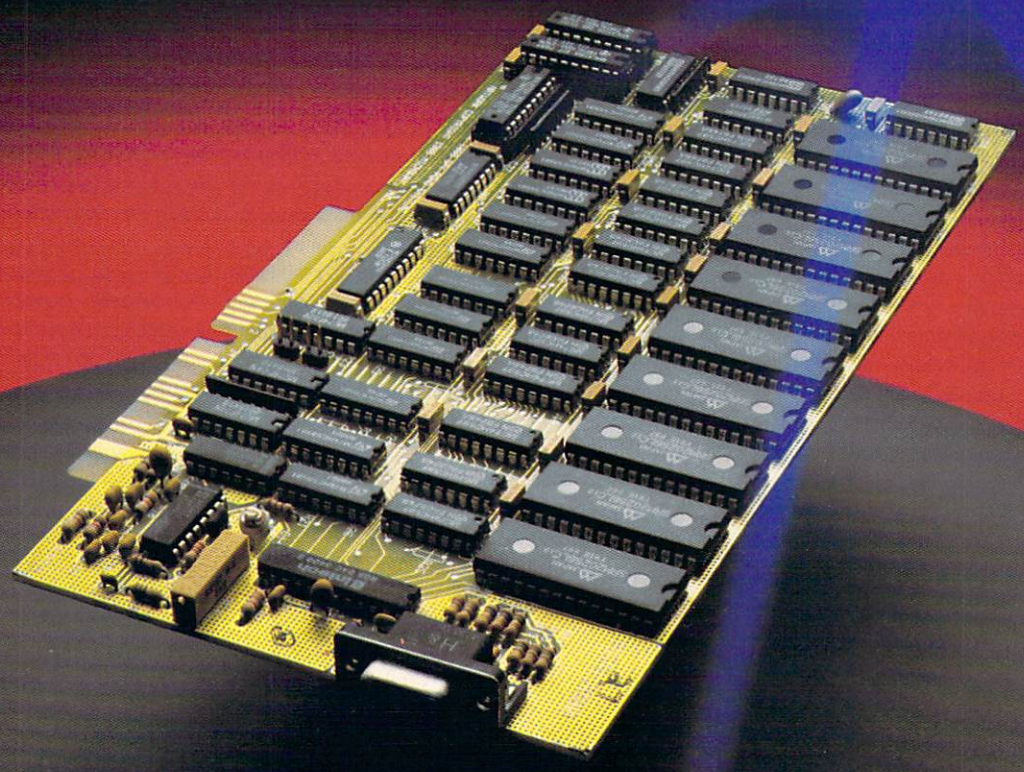
Airball—a different kind of basketball simulation.



(left)
Design 3D from Gold Disk

(right)
Designer Databases from Software Visions

AGA 2000
The flickerFixer™



UNLOCK THE GRAPHICS POWER OF YOUR AMIGA 2000!

flickerFixer is an advanced graphics adaptor that eliminates your Amiga 2000's interlace flicker and visible scan lines. The result: superior quality color or monochrome graphics and text — for such demanding applications as CAD/CAM, Desktop Presentation, Graphic Design, Animation, 3D Modeling, Video, and Word Processing.

flickerFixer upgrades the Amiga 2000 with a flicker free 4096 color palette, has an overscan mode that features a screen size of 704 x 470 pixels and drives most of the popular PC Multiscan and VGA monitors, including the NEC Multisync and Mitsubishi XC1429C.

Accolades include: ***Best of 1988 Award,*** Commodore Magazine (12/88); ***#1 Reader's Choice Award***, AMIGAWORLD (12/88); ***"The display is fantastic . . . It is the best display we have ever seen on any computer system."*** Amiga GURU (5/88)

flickerFixer fits into the Amiga video slot, is fully compatible with all software, and does not modify the standard Amiga video signals. For more information or to order, call MicroWay Sales at (508) 746-7341 or your Amiga Dealer. Priced at $595, ***flickerFixer*** is made in the USA and is FCC Class B approved.

MicroWay . . . Respected throughout the industry for high quality engineering, service and technical support.

MicroWay

World Leader in PC Numerics

P.O. Box 79. Kingston, MA 02364 USA (508) 746-7341
32 High St., Kingston-Upon-Thames, UK, 01-541-5466
USA FAX 617-934-2414 Australia 02-439-8400

In recent years there have been two major leaps forward in personal computing—HyperCard and Multitasking. Bill Atkinson's brainstorm combined the functions of database, word processor, and paint programs in a streamlined, stand-alone package the user could navigate freely through Hyper-Text. Suddenly the average user could create customized links between picture, sound and text data, effectively personalizing his computer environment and allowing access to a simplified form of programming, albeit with a certain expanded overhead.

Multitasking was, and is, Amiga's ace in the hole. With your 500, 1000, 2000 or 2500 you can simultaneously run the same applications HyperCard emulates. The IFF picture and sound formats allow data created in one vendor's product to be modified, combined, and shared with other programs for image processing, desktop publishing, animation, business presentation, etc. HyperCard's advantage is the transparency of data movement around its inner pathways. This ability has its costs: each part of HyperCard is somewhat limited by the need to keep the overall code size manageable. If the Amiga user could automate the direct transfer of data between several full-blown multitasking applications, the effect would be one powerful super-application. Enter ARexx.

You have probably heard a great deal about ARexx over the past few years since author William S. Hawes developed his implementation of M.F. Cowlishaw's REXX language, an IBM procedures language for use with mainframes. Recently the Amiga community has seen a veritable blizzard of ARexx-compatible programs in almost every software area. ARexx-compatible programs now available include:

- CygnusEd Professional and TxEd+ commercial word processor/editors
- Uedit and QED shareware editors
- AmigaTeX typesetting program
- C.A.P.E. assembler
- Microfiche Filer Plus and Superbase Professional databases
- Superplan and Plan/It spreadsheets
- Atalk-III and VLT terminal communications programs
- Lattice compiler
- Wshell command shell
- Nag Plus 3.0 schedule assistant

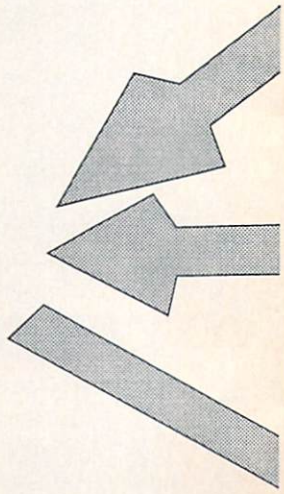
Soon to be, if not already, available programs include:

- Digi-Paint 3 paint program
- Turbo Silver 3.0 SV, PageRender 3D, and 3D Professional 3D ray-tracers
- FrameGrabber II software
- BaudBandit terminal communications program
- Magellan artificial intelligence/expert system and companion Toolkit Interface release
- The Advantage spreadsheet
- Thinker hypertext idea processor
- Atredes 1.1 and BBX BBS programs
- FreD speed-dialer
- Designer Database series releases for use with Microfiche Filer Plus

Certainly an impressive list, it will continue to grow as each new program or update with ARexx adds power and functions to every program that talks ARexx. Though now past the million-shipped mark, the installed base of Amigas is small compared to its more established predecessors.

And the economic rewards for software developers are naturally more limited and proscribed. Productivity software is therefore written for the greatest common denominator, and the user must either adapt to a program not entirely compatible, or hire a programmer to write an application designed to the user's specifications. Also, multitasking issues require new concepts and solutions that are just now being devised on the Amiga.

Adventures in ARexx



by Steve Gillmor

These sophisticated programs lean heavily on macros to allow the user to manually create appropriate strategies and techniques and then record them for automatic playback. ARexx steps into this ever-growing complex environment and offers a common language for these macros, allowing the use of the same shortcuts from any ARexx-compatible program.

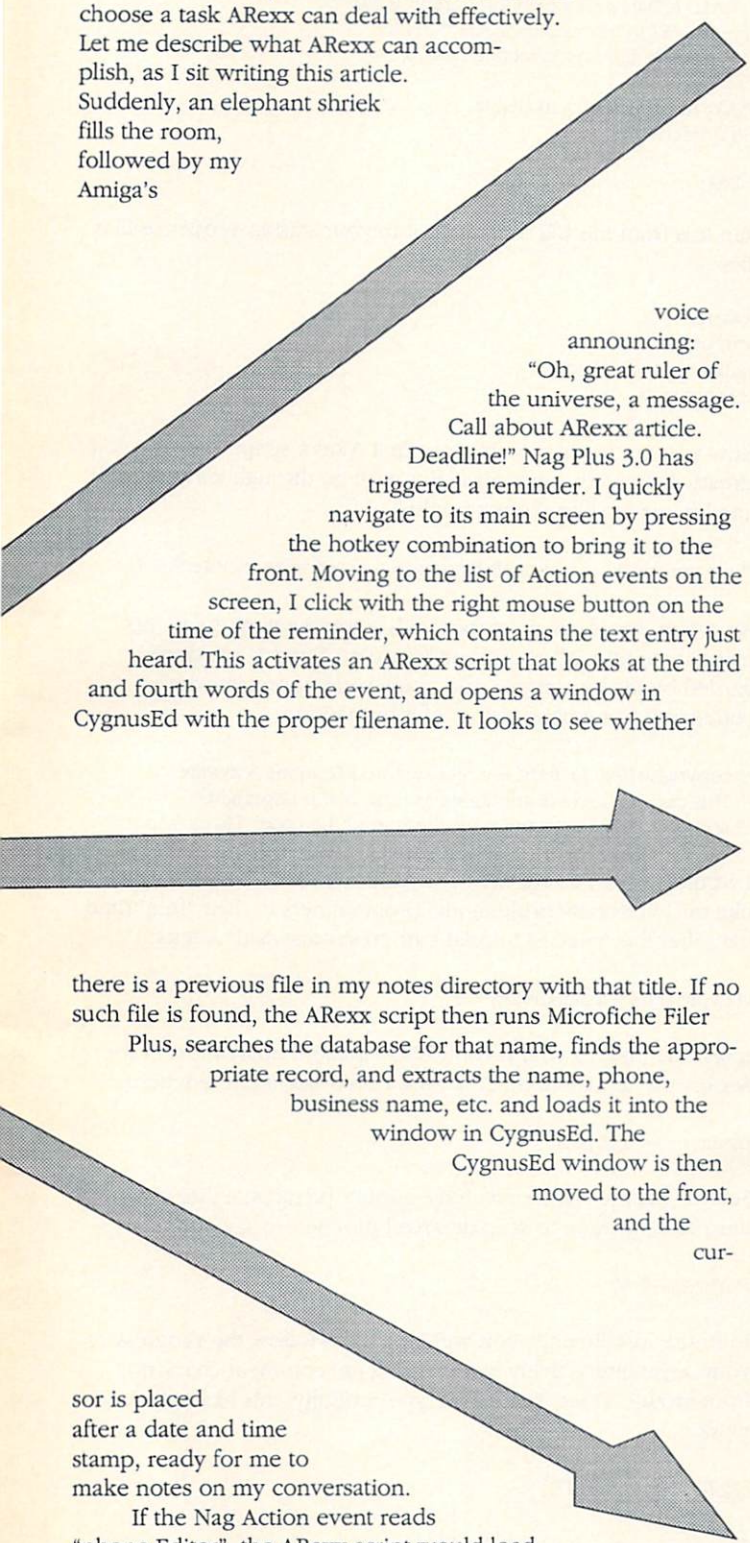
But I'm sure you've heard enough about how useful and flexible ARexx is. What's less clear is how the average Amiga user can get into and master this language. Although Bill Hawes describes ARexx as "an easy-to-learn yet powerful language", he is speaking to the novice programmer, not the user. The ARexx manual is full of information and very useful once you've gotten started. But it is designed as a reference tool, not a tutorial. Some programs that rely heavily on ARexx, like Microfiche Filer Plus, contain considerable information on implementing ARexx, but you can use the macros without understanding how they work, or how to apply these scripts to other programs. Some

programs refer to ARexx on the box but do not mention it in the manual, just a readme file on the disk. Others, like CygnusEd Professional, have elaborate documentation, but the choices and complexity of configuration decisions can bewilder the average user.

The best way to hurdle this Catch 22 is to choose a task ARexx can deal with effectively. Let me describe what ARexx can accomplish, as I sit writing this article. Suddenly, an elephant shriek fills the room, followed by my Amiga's voice announcing: "Oh, great ruler of the universe, a message. Call about ARexx article. Deadline!" Nag Plus 3.0 has triggered a reminder. I quickly navigate to its main screen by pressing the hotkey combination to bring it to the front. Moving to the list of Action events on the screen, I click with the right mouse button on the time of the reminder, which contains the text entry just heard. This activates an ARexx script that looks at the third and fourth words of the event, and opens a window in CygnusEd with the proper filename. It looks to see whether there is a previous file in my notes directory with that title. If no such file is found, the ARexx script then runs Microfiche Filer Plus, searches the database for that name, finds the appropriate record, and extracts the name, phone, business name, etc. and loads it into the window in CygnusEd. The CygnusEd window is then moved to the front, and the cursor is placed after a date and time stamp, ready for me to make notes on my conversation.

If the Nag Action event reads "phone Editor", the ARexx script would load FreD, the speed-dialer program, and dial the appropiate number through the modem. But in

(continued)



this case, I want to collect my thoughts and get my excuses straight before calling, so I just have Nag remind me. I click into the other window on the CygnusEd screen and continue working on my article.

Together with Nag Plus author Richard Stockton, and with the support of CygnusEd's Perry Kivolowitz and Microfiche Filer's Gary Samad, we slowly refined this system. Now that the system is stable and in use on several Amiga configurations, from a 1000 with 1 meg and 2 floppies to a 3 meg 2000 with an 80 meg GVP/Quantum hard drive, we are modularizing the ARexx scripts so the system can expand and configure easily to whatever ARexx-compatible editors, databases, terminal programs, etc. are available.

Although these modules are intricately interwoven, they are direct descendants of the first scripts we built by modifying existing examples. So let's look at one of these original fledgling programs, an ARexx example that uses any public-domain picture viewer to display images from within any other application.

Before we look at viewpic.rexx, let's make sure our environment is set up correctly. Assuming you have followed the manual's instructions for installing the ARexx:libs directory contents in your system LIBS: directory (as well as the same operation for the c: directory), you will need to add two other libraries to make our viewer and other scripts work. Available on Plink and other pay networks are two required files—Rexxarplib1.3 and Arp1.3. When you have downloaded and unarced or unzooed them, you will find rexxarplib.library and arplib.library, which you should also copy into your system LIBS: directory. Now open an editor or use ED to enter this ARexx script, and save it as setup.rexx in your rexx: directory.

```
/* setup.rexx - Mounts the REXX support libraries */
  SAY 'Mounting rexx support libraries
...' IF ~SHOW('L',"rexxsupport.library") THEN
    CALL ADDLIB "rexxsupport.library",0,-30
 IF ~SHOW('L',"rexxarplib.library") THEN
    CALL ADDLIB "rexxarplib.library",0,-30

  SAY 'REXX Libraries available...'
 SAY SHOW("L")

   EXIT
```

Run this from the CLI or include it in your startup-sequence like this:

```
rexxmast
waitforport rexx
assign rexx: dh0:rexx
rx setup.rexx
```

Now you are ready to write your first ARexx script. Actually you already have with setup.rexx. But we'll go through viewpic.rexx line by line to show how it works.

```
/* viewpic.rexx  -  uses ARP file requestor to display picture file  */
```

All ARexx programs must start with a comment, and why not make it the title. All characters between /* and */ are disregarded by the interpreter, and you'll soon appreciate the comments as a crucial source of information.

```
/* copyright 1989 Richard Lee Stockton and Gramma Software. */
/* This code is freely distributable as long as this copyright */
/* notice remains, unchanged, at the start of the code. Thank you. */
```

I would suggest doing what the man says, because it's scripts like this written by professional programmers in their "free" time that offer the quickest tutorial path to success with ARexx.

```
viewprogram = 'c:Superview'
```

Insert the path to your favorite show-picture utility here. (I use Superview, available on Plink, the Fish disks, and elsewhere).

```
picdir      = 'dh1:pics'
```

Set this path to where you have a lot of pictures, either on a hard drive or ram: to scan de-arced pics before saving them.

```
/* TRACE ?R */
```

In future installments, you will be able to follow the progress of your script interactively by removing the comment characters from around Trace. But if you type carefully, this example will work.

OPTIONS RESULTS

This is a toggle that switches function returns between error codes and character strings. Here we are enabling the function Getfile to return a filename string. Getfile? Don't worry, it's coming. Read on.

```
IF ~SHOW(f,STDOUT) THEN CALL OPEN(STDOUT,'NIL:')
/* Guru Insurance */
```

This mouthful provides a default output stream if you are running a command that requires an output stream.

```
filename  = ""
```

Two double quotes read as a null or empty string. We assign a null value to filename so the program does not think there is a filename named "filename".

```
filename = GETFILE(150,36,picdir,"","  Select A Picture File To Display ")
```

This line opens a file requestor at 150 pixels right and 36 pixels down from the upper lefthand corner of the Workbench screen. The requestor displays the contents of where you have assigned your picture directory. Another blank string follows since we have yet to select a filename, and the final entry contains the text information displayed in the title bar of the requestor.

```
IF filename=="" THEN EXIT 5
```

This is to safeguard against the possibility of clicking on "Cancel" in the requestor. If you do, the program quits with only the mildest (level 5) of complaints.

```
fileinfo = STATEF(filename)
IF WORD(fileinfo,1)~='FILE' THEN EXIT 10
```

The above lines make sure the filename result is indeed a filename, and not a directory or a device like dh1: or ram:. The ARexx function STATEF returns a character string containing several words, the first of which is either DIR (for directory) or FILE, or empty, as in blank, zip, nada. The next line checks that first word with the ARexx function WORD. If that word is not equal to (~=) FILE, then sayonara.

```
ADDRESS COMMAND viewprogram filename
```

If we have made it past this gauntlet of error checking, it's time to show the file, and display our picture. ADDRESS COMMAND is the language ARexx uses to RUN a DOS program. It uses the viewprogram you've assigned, and the filename you've selected and checked as arguments. Voila

```
EXIT
```

Although not required here, EXIT will be necessary when we begin expanding this and other routines, so we put it in anyway.

Here, without my narration, is the complete script:

```
/* viewpic.rexx - uses ARP file requestor to display picture file */

/*  copyright 1989 Richard Lee Stockton and Gramma Software.   */
/*  This code is freely distributable as long as this copyright  */
/*  notice remains, unchanged, at the start of the code. Thank you. */

viewprogram = 'c:Superview'
picdir      = 'dh1:pics'

/* TRACE ?R */
OPTIONS RESULTS

IF ~SHOW(f,STDOUT) THEN CALL OPEN(STDOUT,'NIL:')
/* Guru Insurance */

filename  = ""

filename = GETFILE(150,36,picdir,"","  Select A Picture File To Display ")

IF filename=="" THEN EXIT 5

fileinfo = STATEF(filename)
IF WORD(fileinfo,1)~='FILE' THEN EXIT 10

ADDRESS COMMAND viewprogram filename

EXIT
```

Save in your rexx: directory as viewpic.rexx and run from the CLI by typing (without the quotes) "rx viewpic.rexx". If you're like me, and have Superview in your c: directory and IFF images in dh1:pics on your hard drive, you will be presented with a requestor and a choice of pictures when you hit <return>. Click on one, then on OK, and bingo. You can also load this script in as a macro (I have it as F1 when in CygnusEd) and run it from within any ARexx-speaking program.

One more quick example:

(continued)

```
/* sound.rexx  - uses ARP file requestor to play soundfile  */

/*  copyright 1989 Richard Lee Stockton and Gramma Software.  */
/*  This code is freely distributable as long as this copyright  */
/*  notice remains, unchanged, at the start of the code. Thank you. */

playsound = 'c:Sound'
sounddir  = 'dh1:sounds'

/* TRACE ?R */
OPTIONS RESULTS

IF ~SHOW(f,STDOUT) THEN CALL OPEN(STDOUT,'NIL:')
/* Guru Insurance */

filename  = ""

filename = GETFILE(150,36,sounddir,"","," Select A Soundfile To Play ")

IF filename=="" THEN EXIT 5

fileinfo = STATEF(filename)
IF WORD(fileinfo,1)~='FILE' THEN EXIT 10

ADDRESS COMMAND playsound filename

EXIT
```

If you look carefully, the only difference between viewpic.rexx and sound.rexx are the obvious changes of path names, program variables, and comments. Just make those changes and load a sound player into your c: directory alongside your picture viewer. Save this new version as sound.rexx in your rexx: directory and you're set. Not only can you learn from examples, but you can reuse them in total or in parts to create other scripts with ease and minimal typing. And if it works, don't fix it.

Once you've gotten over the hump of getting started, ARexx will begin to make more and more sense. The language is written with a small number of commands, most of which are English-like and self-explanatory.

There are many listings of examples on PeopleLink, GEnie, and CompuServe, as well as Bix and local BBS's. Bill Hawes answers questions E-Mailed to him on the above-mentioned nets, and he's just begun an ARexx class the first Tuesday of every month on Plink, beginning on Line 99 at 9:30PM EST, 8:30PM Central. If you miss the class, you can find a transcript in the library.

And don't forget to read the manual; you'll be amazed how things that just a few days ago seemed totally incomprehensible are suddenly so obvious. The glossary in the back is short but useful and, if you cannot find a term there, check the index. You will usually be directed to a definition in the text.

As more and more developers incorporate ARexx into their products, the number of example scripts and business-oriented macros will grow exponentially. The overhead of adding ARexx ports to a program is minimal, and justified by the increased utility and maximized use of multitasking. Indeed, the ability to automate and stitch together into a seamless fabric the wide variety of graphics and animation software is vital to the success of the Amiga as a serious professional tool. Stay tuned.

•AC•

NAG PLUS 3.0

SCHEDULE ASSISTANT

review by Marion Deland

Nag Plus 3.0 is a useful little program that takes advantage of the Amiga's multitasking abilities. It calls itself a Schedule Assistant, and combines an appointment calendar, a "do list" and a telephone dialer. And it talks.

This is a cute program. It might be a little too cute, but you can load it from your startup-sequence and bypass some of the cuteness. The name of the company is Gramma Software, for instance, and the program icon is a sweet old lady shaking a finger: "For your own good." The author is Richard Lee Stockton, and the manual tells us his favorite books ("Hitchhiker's Guide") and pets ("imaginary"). But cuteness aside, this is a useful program that does what it claims to do at a reasonable price.

The basic appointment calendar was first introduced as the public domain program Nag, available on Fred Fish Disk 161. There are major improvements in the commercial version, including Notepad access, ARexx support, and phone dialing. As I write this, version 3.1, with extended ARexx support, is planned for a June 1 release. Stockton promises an inexpensive upgrade policy for registered owners.

When you first run Nag Plus, it hides on your screen as a tiny window which expands when you click on it or use a "hot key" combination of key-strokes. When it expands, the first thing Nag Plus does is check the "year files" and let you know if you've missed any reminders. (More about that later.)

Then the Main Window appears. It shows the appointment calendar, starting the current date. Only days with scheduled appointments are displayed; and one of the many nice features in this program is that current dates (if shown) are clearly marked "today" and "tomorrow". A code of colored dots tells you if you have asked Nag Plus to give you a reminder by voice, sound or screen message.

At right is the overall calendar showing the current month, with the current date marked. The time is also displayed. A click of the mouse button changes the current month, day, or year.

From the main window, you can "check off" a completed appointment by clicking on it. It still appears, but in the background. Clicking on the Checkmark button makes it disappear entirely from the main window.

You can print out the appointments for one day or several days. If your printer has a small font, checked-off appointments will print out in that font, and scheduled appointments in regular type. Nice!

You can search the calendar for appointments that contain a specific word or phrase. You could use this to find all meetings with a particular person.

Nag Plus will even dial your calls for you. Just enter the phone number in your appointment and click on it when

"Nag Plus 3.0 combines an appointment calendar, a "do list", and a telephone dialer. And it talks."

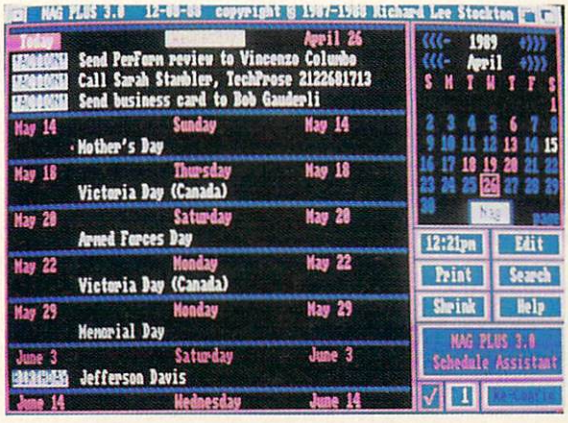


***Figure One**: The main window appears, showing the appointment calendar. At right is the overall calendar showing the current month.*

you want Nag Plus to make the call. (You need a Hayes-compatible modem that will transmit at 300 baud.)

Something I like very much is the ability to attach a text file to any word or date, using either the Amiga Notepad or your favorite text editor. But there is no marker of any kind to tell you a text file is attached. I think this would be useful.

A Shrink button shrinks Nag Plus back down to the tiny—and I do mean tiny!—window in the top left corner of your Workbench screen.

To add, change or delete events, switch to the Edit Window. Enter events, one-per-line, in the edit box. You can schedule birthdays, holidays, anniversaries or deadlines, as well as regular appointments and "action" events (the Do List). You can get around the one-appointment-per-line limit by typing in several lines at one-minute intervals, then having Nag Plus display the times only at 5 or 15-minute intervals.

It took me a while to figure out how to delete something from my Do List. ("Action" items, including those that are checked off, are transferred from day to day until you delete them.) I looked for help in the manual, but there was just a passing mention of deleting appointments, and no reference to it in the index. Eventually, I found the delete button. I discovered, too, that you can keep a record of completed actions by entering the time you completed them, then checking them off. That way, they become an "appointment" and stay put, but in the background.

You can also ask Nag Plus to remind you about appointments. The program is appropriately named—unless you set it for a once-only reminder, it repeats the reminder again and again until you click to acknowledge it. The first time this happened to me I was on the telephone. Try explaining to a client that the peculiar voice announcing at 30-second intervals, "Oh, great ruler of the universe! Nag has a message for you." is actually your computer! (That's the default message, by the way. See what I mean about cute?)

You can ask to be reminded at the time of the appointment, or ahead of time, at intervals ranging from 15 minutes to 30 days. That's where the "missed nags" message comes in.

Suppose your parents' anniversary is on September 11, and you promised the family you'd plan the party. You asked Nag Plus to remind you about it 30 days ahead—on Saturday August 12th. But you were away for the weekend, and didn't turn the computer on until Monday August 14th. Nag Plus will tell you about the "missed nag". But if you go away on vacation, and don't turn on your computer until September 13, you've missed the anniversary altogether, and Nag Plus won't bug you about it.

If this starts to sound confusing, take heart; comprehensive help is available. Clicking on the Help button in the main window brings up a Help window which explains each area of the screen.

The manual, although well-designed and spiral bound, is not as clear as it could be. But a lot of thought has gone into it: there are sections for new Amiga users and advanced users, as well as detailed instructions on configuring Nag Plus to suit your needs.

There are also a couple of disk files containing "notes" and "hints"—mostly information on executing REXX and EXEC files and attaching notes to appointments. (What exactly does "prepended to the event-text" mean?)

Configure it your way

Nag Plus is very flexible. For example, you can schedule appointments at one-minute intervals. Many appointment calendars available for other computers lock you in to 15-minute or half-hour segments.

You can reconfigure Nag Plus at any time through a separate SetNag program. Among other things, this lets you change the greeting message, the voice (pitch, rate, etc.), and sound. The Nag Plus disk includes a number of digitized sound files, including a fanfare, an elephant and a sneeze. (You can hear them by clicking on their icons in the Sounds drawer—good idea!) You can also use the standard Amiga beep.

More than one person can use Nag Plus as it will maintain separate schedules and "tiny window" configurations for each person. You can tell it where to look for the year files, note files and help files. You can change the size of the buffer that contains the year files. You can disable many of the features, like repeating NAGs, and missed NAGs, and you can change the characteristics of the tiny Nag window, reverse Workbench colors, and change the time format. This program was definitely designed to be flexible.

Incidentally, you can execute REXX (and ARexx) and EXEC commands from within Nag Plus, by entering them as appointments. Richard Stockton showed me how he used this feature to back up his files from RAM to floppies automatically. There are suggestions for this in the manual.

Nags about Nag

Some things about Nag Plus could be improved. I didn't like the way the manual describes everything by color—orange box, white dot, etc. This is fine if you use the original Workbench colors, but I don't. According to Stockton, a printer's error left out an explanatory page, but to my mind this isn't really the answer. I don't want to have to translate constantly as I learn a new program. It would have been better if symbols were identified by shape rather than color.

I would like Nag Plus to be able to maintain expense records, because most people keep track of expenses in their appointment calendars, with the appointments. While Nag Plus allows you to record them through Notepad, there is no way to total them, which I would find useful. But I realize that a choice sometimes must be made between extra features and the memory they take up.

I like Nag. I load it from my startup-sequence so it sits quietly in a corner of my Workbench all the time, and I don't have to see that silly old-lady icon. I use it mostly as a Do List, to keep track of phone calls and deadlines. And I'll learn how to use it with ARexx—real soon now.

•AC•

Gramma Software
17730 15th Avenue N.E., Suite 223
Seattle, Washington 98155
(206) 363-6417
Nag Plus 3.0, $79.95

Digi-View Gold

review by Bruce Jordan

Once in a great while a product comes along that makes my job as a reviewer a complete joy—a product so interesting and fun you'd have to pry it out of my hands to get it back! Such is the case with NewTek's video digitizing system, Digi-View Gold. Not only did this unique product reaffirm my belief in the Amiga,but it also brought back that sense of excitement I experienced when I fired up my Amiga for the first time!

I think if I let it, my Amiga's synthesized voice would be croaking out a line from that old hymn, Amazing Grace: "I once was blind, but now I see...", because that's exactly what the Digi-View Gold system does. It allows your Amiga to see and capture full color images of anything you can place in front of a video camera.

I really hate stooping to that tired old marketing cliche, but the possibilities for Digi-View Gold truly are limited only by your imagination. Artists can input their drawings directly from paper—no more drawing with Mr. Mouse! Graphic artists can stuff disk after disk with bits and pieces of the real world to be chopped up and manipulated as they please. Video-ites can now add their Amiga to the ranks of their video ensemble. If your bag is astronomy, why not hook up a video camera to your telescope, digitize the night sky, and then use Digi-Paint to pull out the detail

If you're into robotics, Digi-View Gold gives your Amiga sight! And since Digi-View Gold uses standard IFF graphics files, hackers can use their digitized images with their own programs, so we may see a change in the look of video games in the near future.

The list goes on. There's even a way of creating three-dimensional images using Digi-View Gold! However, before we get carried away, let's take a closer look at the Digi-View Gold system, what it contains, what it does, and how to use it.

The basics

The basic system consists of four components: the Digi-View Gold package, a video camera, fluorescent lights (Digi-View is color-balanced for fluorescent lights), and some sort of copy stand or tripod.

The Digi-View Gold package contains a video input box that connects a camera to your Amiga, the Digi-View Gold software, a three-color filter wheel, and a metal bracket for mounting the filter wheel to a video camera. As for the camera, you'll want the recommended Panasonic WV-1410, closed-circuit, black-and-white video camera (or its equivalent). Further, the copy stand offered by NewTek is adequate, but I think you could do better—both in price and quality—at your local camera store or through a mail-order house that specializes in photographic equipment. The fluorescent lights are your responsibility and can be purchased at almost any hardware store.

Those of you with home video cameras may be wondering why I've included the Panasonic camera into the basic system. "Why not just use my own camera?", you might ask. Good question. The answer is that, while Digi-View does make allowances for the use of home video cameras, you may run into a number of problems.

First, your home video camera does not have the resolution of the Panasonic camera (better than 550 lines!). Your camera's lack of resolving power may make it impossible to render a quality color image using Digi-View. Tracking errors is another problem home video camera users may run into. Digi-View expects a 2:1 sync ratio. If your camera uses random sync, or if it

(continued)

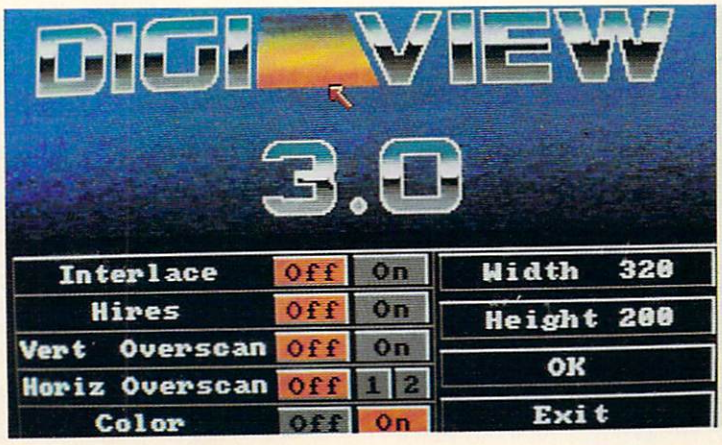


Figure One *(bottom left)*
On the title screen will be several gadgets that give you choices regarding resolution, overscan, and whether to digitize in color or black and white.

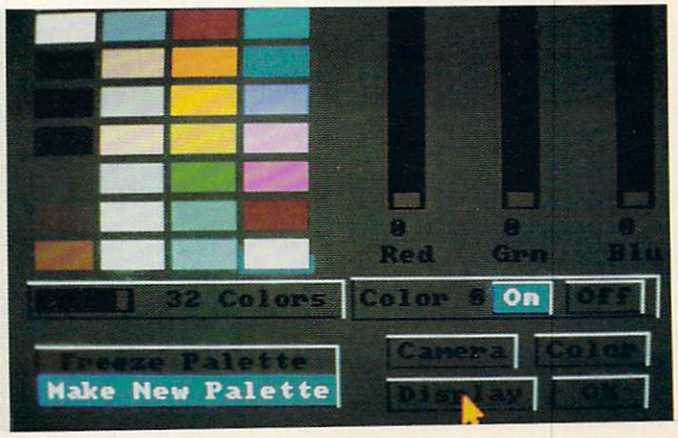
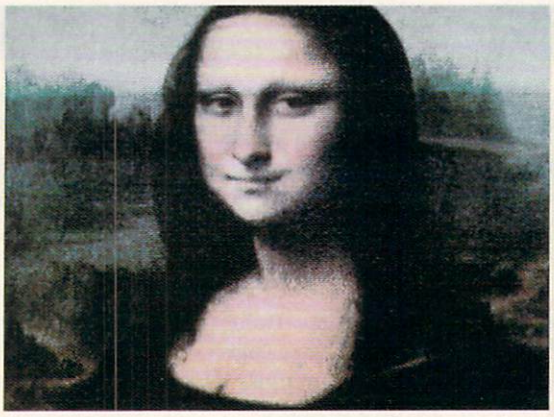
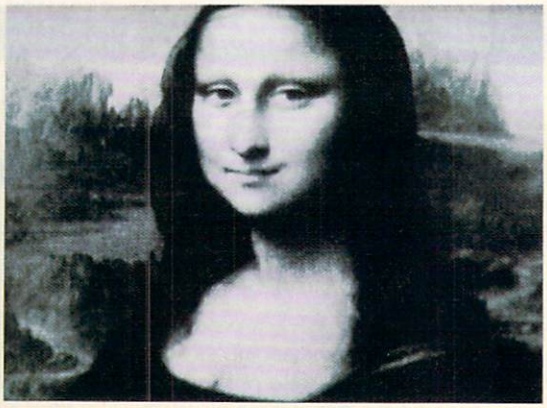


Figure Two *(bottom right)*
"Palette"displays the palette of colors being used by the current image. It allows you to alter the individual colors in the palette.





Figures Three & Four *(left & right)*

Sample digitized screens of the Mona Lisa, digitized with DigiView

produces a sloppy 2:1 sync signal (not uncommon), you may wind up with tracking errors that result in a choppy, color-distorted picture. NewTek calls this the "jaggies." Finally, even though your camera can see in color, Digi-View Gold uses a three-color digitizing process dependent on the filter wheel. You may find that the filter wheel does not fit your camera, as is the case with the Sony CCD V3 camcorder. If you're not handy with tools, you'll probably wind up holding the filter wheel in front of your camera's lens. This gets old real fast! Therefore, to get the best possible images with the greatest ease, your best bet is to go with the Panasonic black-and-white camera. It's inexpensive and well worth the investment.

Figure Five *(bottom left)*
Color also displays a number of slider switches giving you control over brightness, contrast, color saturation, sharpness and the overall red, green, and blue aspects of the picture.

Figure Six *(bottom right)*
Another feature offered on the "Camera" window is a choice of capture modes.

Paint by numbers

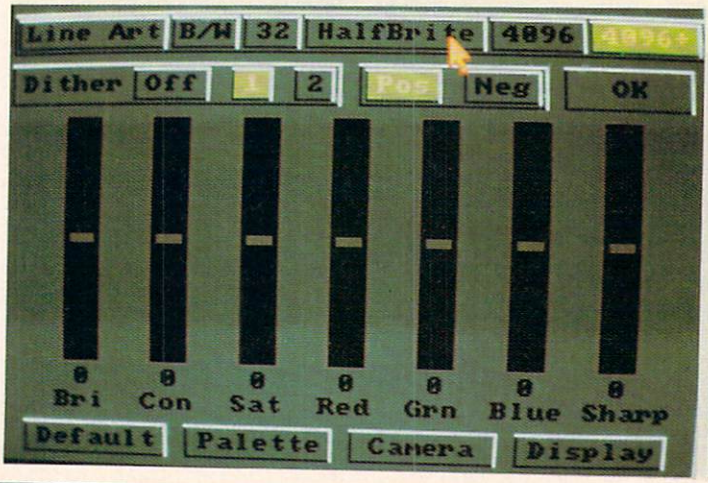
With the basic system, digitizing is surprisingly simple. First, make sure the power to your Amiga is off. Then plug the video input box into the parallel port of your 500 or 2000 (note: Amiga 1000 users will need a "gender changer" to convert the parallel port from male to female). Next, connect your video camera's video output to the phono-type socket in the back of the Digi-View video input box. Now turn on the power to both your Amiga and the camera. After you Kickstart and your Amiga requests the Workbench, insert the Digi-View Gold disk. Double click on the icons and the title screen will appear. On the title screen will be several gadgets that give you choices regarding resolution, overscan, and whether to digitize in color or black and white.

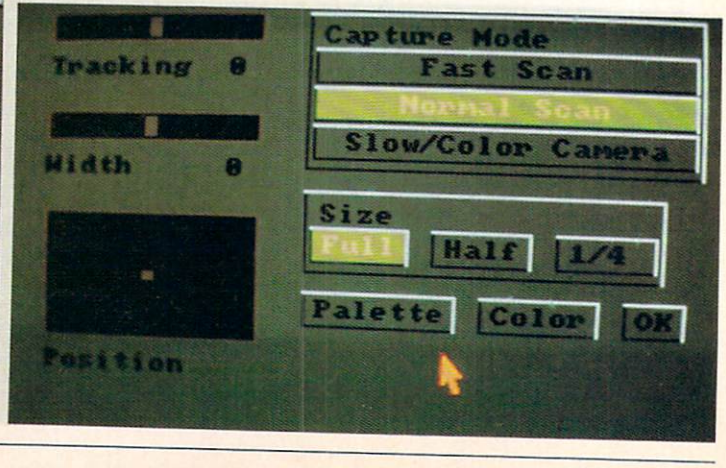
Your choices for horizontal and vertical resolutions range from 320 pixels to a whopping 780 for the horizontal, and from 200 to 480 pixels for the vertical. Each of these unprecedented upper limits for resolution is achieved by overscan, which simply means that the digitized image appears over the entire screen without a border.

If you choose to digitize in color, Digi-View defaults to the 4096+ color HAM mode for all resolutions except high resolution (anything at or above 620 horizontal pixels). However, once inside Digi-View, you can opt to use fewer colors—say, thirty-two. This allows you to use your digitized images with paint programs such as DeluxePaint.

In high resolution, Digi-View defaults to sixteen colors and does not allow the use of HAM mode. But fret not. With its powerful graphics routines, Digi-View does an excellent job of extending the apparent color range, producing images of unbelievable color and clarity, even with as few as sixteen colors.

From this point on, using Digi-View Gold is a matter of available RAM. This is especially true for those using Amiga 1000's with the standard 512K of memory. Users of unexpanded 1000's can expect to be limited to two modes—black-and-white, with a maximum resolution of 320 X 400, and color, with a resolution of 320 X 200. To make matters





worse, color work on an unexpanded 1000 is possible only if the external disk drive is disconnected. "WHAT?" Yep, it seems the drive uses precious RAM required by Digi-View. Therefore, unexpanded 1000 users can figure on a one-drive system. Go on, it ain't that bad! Digi-View Gold does an excellent job of reproduction even in limited low-res. I guarantee you will not be disappointed. Besides, it'll be just the excuse you need to get that RAM expansion you've been wanting.

For this review I used a system consisting of an Amiga 2000 sporting a hard drive, three meg of extra RAM, the new Flicker Fixer board, and an NEC Multi-sync, high-resolution monitor. (Yeow!) The system gave Digi-View Gold all the room it needed to run and allowed me to push the product to its limit. The results were absolutely breathtaking! It's a shame that copyright restrictions keep me from showing you all the results, but I can tell you from firsthand experience that the pictures in the Digi-View ads are not exaggerations!

Manufacturing art

For a run-through of the digitizing process, let's say we select low-res (320 X 200 pixels), no overscan, and color mode. We do this by clicking on the appropriate gadgets, and then clicking on the gadget labeled "OK." The program loads, the screen turns completely black, and the mouse pointer turns into a large arrow. This indicates that Digi-View is operating and ready to digitize.

Now suppose we want to capture the image of a pretty woman out of a magazine. Obviously, the first thing we want to do is focus the camera and properly frame the image. The people at NewTek say it's safe to unplug the camera from the back of the Digi-View input box and plug it into your monitor's composite video socket while everything is running. This allows you to position and focus the camera in real time, without having to wait for the digitizing process. I did this repeatedly and it seemed fine.

Now that the image is in focus, framed, and we've positioned the lights to get rid of any glare, plug the camera back into the Digi-View input box. We are now ready to shoot.

While placing the red filter in front of the camera's lens, hold down the right mouse button, and a menu bar will appear at the top of the screen. It will display three menu titles—PROJECT, DIGITIZE, and CONTROLS. Point to DIGITIZE and a menu will drop, displaying three choices—"Red", "Green", and "Blue". Point to "Red", release the mouse button and, suddenly, vertical sections of the picture about a quarter of an inch wide will begin to appear on your monitor's screen.

When the image is complete, a smoothing process will begin at the top of the screen and work its way down to the bottom. The entire process takes about a minute. Although the image you have produced is in black and white, you have just digitized the red portion of the woman's picture. Do the same for "Green" and "Blue", each time making sure that the corresponding color filter is in front of the camera's lens.

Now that we have all three color images stored in memory, we must give Digi-View the command to combine these images into a single color image. To do this, pull down the CONTROLS menu, and select "Display." The screen will go black for a moment, and then the full 4096+ color image will begin to appear, line-by-line, from the top of the screen to the bottom. And there we are—our first digitized image. Now pull down the PROJECT menu and save the picture to disk before someone comes in and catches us!

Under Controls

Once you have something digitized, Digi-View gives you quite a bit of control over various aspects of the image. The PROJECT menu allows you to load and save images, load the palette of one image for use with a different image, and display histograms of the red, green, and blue aspects of the current image on the screen.

The CONTROLS menu has four choices: "Color," "Palette," "Camera" and "Display." We've already seen what Display does. Its function is simply to display the current image in memory. However, each of the other options on the CONTROL menu bring up gadget windows that provide a surprising number of features for controlling the digitizing process, as well as allowing you to alter your digitized images.

Choosing "Color"from the CONTROLS menu brings up a gadget window that allows you to select various color formats. For instance, you have a choice of Line Art (which gives a high contrast version of the image), black and white, thirty-two colors, 4096 color HAM mode and an extended 4096+ color HAM mode. Color also displays a number of slider switches giving you control over brightness, contrast, color saturation, sharpness and the overall red, green, and blue aspects of the picture.

"Palette"displays the palette of colors being used by the current image. It allows you to alter the individual colors in the palette and even change the number of colors used in the digitizing process. The default palette may display anywhere from sixteen to sixty-four colors, depending on the resolution and color mode of the current image.

"Camera"gives you control over certain aspects of the video camera. For instance, there are two controls, Width and Position, which allow you to change the apparent width and positioning of the image without having to physically move the video camera. There is another control labeled Tracking that allows you to adjust the tracking between Digi-View and your camera.

Yet another feature offered on the Camera window gives you control over the size of the digitized image. You have the choice of digitizing a full, half or quarter screen image. This allows you to obtain smaller images without the resolution loss typical of the size-reduction routines used in most paint programs.

Conclusion

Overall, Digi-View Gold is a fantastic, easy-to-use package, adding more fun to the Amiga than any product I've seen in a long time. With the right camera, the images produced by Digi-View Gold push the Amiga to the limits of its graphics abilities. At times, the color of the digitized image was just a little off, but I suspect that the more one uses the product, the better one would become at fine tuning such adjustments. Other than this minor difficulty, the software performed without a hitch. To add further praise, NewTek brought the whole system in for a very reasonable price. In fact, it's surprisingly inexpensive for the Amiga market!

So whether you are interested in art, video graphics, family snaps or the great nebula in Orion, Digi-View Gold is definitely a winner. With Digi-View Gold, the world is yours to capture.

•AC•

Bug Bytes

by John Steiner

The release of the Amiga 2500 and the popularity of the 2000 HD have exacerbated problems with the A2090 controller card and high-resolution, 16-color overscan. The problems are occurring with some lack of consistency; some people have a problem, and others do not. Programs that make heavy use of chip RAM in overscan mode, like Deluxe Productions, Pro Video Plus and other high-resolution applications, create a problem for the A2090 controller card. What makes the problem worse in some machines, and virtually non-existent in others is a mystery. Some people make changes to their startup-sequence, and the problem disappears. Others make the same changes, and the problem remains. If you have a problem with your A2090 in high-resolution 16-color mode or high-res overscan, try the following suggestions.

If you have expansion memory, check your startup-sequence, and modify it to run SetPatch, then run FastMemFirst before you run binddrivers. In a standard 2000 HD or 2500 startup-sequence supplied by Commodore, these commands are executed in that order. Changing the order of execution will likely make DMA bus problems worse.

You can improve performance during high-resolution and severe overscan operations by specifying a lower transfer rate in the MaxTransfer statement in the mountlist. Unfortunately, this slows down the performance of all other file operations.

Check your MountList for all hard disk fast file partitions and make sure the BufMemType entry is not set to select chip memory. BufMemTypes 0 and 1 select any available memory, BufMemType 2 and 3 select chip RAM, and BufmemType 4 and 5 select Fast Ram.

When the controller begins reading the hard disk, during the screen display of a 16-color or overscan, use Left-Amiga-N to toggle to the Workbench screen during the drive read. Use Left-Amiga-M to toggle back to the application once the file application has completed. If the program uses the front and back gadgets in the top right of the screen, you can use those to select the Workbench and application screens as necessary.

Obtain PATCH2090.ARC, a freely redistributable utility by Khalid Aldoseri that acts as a workaround for problems evident during high-res 4-bitplane screen displays. The program opens and automatically decreases the size of hard disk Read/Write blocks, thus creating less DMA bus contention. You should be able to find it on the information services.

While on the topic of Commodore hardware products, the AT bridge card and the 8 Megabyte Commodore memory card have a problem working together. The problem is caused by the memory map designed into the computer. The 68000 microprocessor has address lines that can handle up to 16 Megabytes or RAM. All but 9 MB are used for the system and custom chips, allowing for the 1 MB internal RAM and the addition of up to 8 more on a memory expansion card.

If you do not have a BridgeBoard, you can install the full complement of memory onto the 8 Meg RAM board. The BridgeBoard, however, requires 2 Megabytes of the address lines before it will function. The effect of this is to drop the remaining available address lines for memory to 7 Megabytes. A full 8 Megabyte RAM card and the AT card cannot work simultaneously in an A2000. The only solution is to either accept less than 8 megabytes on the card, or not use the AT Bridge card. Are there any hardware hackers with a solution to this problem?

The Amiga 2000 is now shipping with one Megabyte of chip RAM. New machines are being supplied with notification regarding the changes required to stock startup-sequences to take advantage of the extra chip RAM. The Setpatch command, which is the first command found in the stock startup-sequence, must be changed to specify the R option, as in the example SETPATCH R. The R option was specifically designed into Setpatch to allow operation with the one megabyte chips.

Commodore has not yet provided dealers with information regarding the field upgrade of 512K chip RAM Amiga 2000's and 500's. It appears that, in addition to a new fat Agnus chip, the computers must be also equipped with Kickstart 1.3 ROM's.

Electronic Arts has released Deluxe Paint III by Dan Silva. The program is a stunning improvement, and its animation features are top drawer. The program lists for $149.95, and upgrades are available from Electronic Arts. For details contact: Electronic Arts; 1820 Gateway Drive; San Mateo, CA 94404; (415) 571-7171.

Deluxe Paint III requires one megabyte of RAM to operate. A major limitation of Deluxe Paint II was its inability to handle large font directories. The new font requester can handle font directories of any size. Also, a string gadget is provided, allowing you to change font directories. The requester then makes the new font assignment for you. If you wish to check its appearance, a SHOW button will display each potential font selection. DPaint III also supports color fonts, and even comes with a small sampling of the beautiful presentation fonts from Kara Fonts.

A friend and I have already used DPaint III's animation capabilities to create a video promotion for our local Amiga Users Group. It performed flawlessly, and the included animation player allowed us to easily transfer the animation to a smaller Amiga for editing onto VHS tape. Animations are saved and loaded in the popular Anim format, which means there are a lot of pre-created animations you can load into DPaint III to help you learn animation techniques. Four animations, including the venerable Half-bright Hill (DPaint III supports 64-color half-bright mode), and a beautiful work entitled Cry are included on the animation sample disk.

A problem in DPaint III's font load system has been reported. When you access a font from a different disk, and then call up the font choice, the program crashes. A technical support representative from Electronic Arts said

they were aware of the problem (which only happens in one-megabyte Amiga systems), and they are working on a fix. The problem is caused by memory fragmentation. The only workaround suggested was to save the drawing, reboot, reload DPaint, reload the picture, and then change the font. A revision to repair this problem may be forthcoming.

Registered WordPerfect users have received the latest issue of "WPCorp Report", which states that WPCorp has just released a "maintenance update" of WordPerfect. A quick call to WordPerfect technical support revealed that the program upgrade has not been released. At the time the newsletter was published, the program was slated for an early April release. A few last-minute changes were called for, which delayed the upgrade release. But by the time you read this, it should be available. Registered users can upgrade for $10.00. If you report a bug in your earlier version to technical support, however, there is no charge for the upgrade.

ARP Release 3 is now available. ARP (AmigaDOS Replacement Project) version 1.2 was a popular accessory for Amiga users who wanted smaller, faster C commands and a more consistent user interface than that provided by the commands on the standard Workbench 1.2 disk. Several version 1.2 ARP commands do not work properly under AmigaDOS 1.3, requiring modification to the ARP command project. The ARP commands are freely redistributable, and can be downloaded from all major information services.

DiskSalv 1.40 has been released. It fixes all known bugs of earlier versions, adds some new features, and enhances some old ones. DiskSalv is designed to read files from a damaged disk volume and restore them to a good volume. It can also restore deleted files, though it is not intended for such use. DiskSalv is more efficient and uses much less memory than DiskDoctor, the C command provided by Commodore for similar functions.

The Roger Rabbit game from Buena Vista has been upgraded. Registered owners have been notified of the upgrade, which makes several improvements, not the least of which is the reduction of game startup from 6 to 3 minutes. Other enhancements have been provided, such as removing the reboot previously required before resuming play. In fact, according to the letter, multitasking with the game is supported.

The upgrade costs $3.75 for registered owners only. If you did not register your software, you will not be able to upgrade. Request for the upgrade must be made on the form provided, and the original disks do not have to be returned. For this reason, I cannot publish the upgrade address. If you wish to upgrade, you must send in your registration card.

That's all for this month. If you have any workarounds or bugs to report, or if you know of any upgrades to commercial software, you may notify me by writing to:

John Steiner
c/o *Amazing Computing*
Box 869
Fall River, MA 02722

...or leave EMail to Publisher on People Link or 73075,1735 on CompuServe.

•AC•

GFA-BASIC 3.0 for the Amiga

$139.95

GFA SYSTEMTECHNIK

Boldly goes where no BASIC has gone before.

- High-Speed Interpreter for easy program development
- Over 300 powerful commands
- FAST! – Execution times comparable to C
- In-line C and Assembler Commands
- Easy access to all Amiga libraries
- Extensive Amiga commands with submenus and built-in file requester
- Built-in Text Editor with syntax checking, procedure hiding and auto-indenting
- 400-page comprehensive manual
- Includes Run-Time Interpreter

Available NOW

from

Antic Software

544 Second Street, San Francisco, CA 94107

Call today to order:

800-234-7001

or see your local Amiga dealer.

(GF9200)

KindWords 2.0

by Marion Deland

In using a word processor with a dot-matrix printer, we've always had to choose between the printer's NLQ font and graphics—we couldn't get both on the same page. That is, until the introduction of The Disc Company's Kind Words. Recently upgraded to Version 2.0., KindWords can combine full color graphics with the printer's built-in font and KindWords' own "Superfonts", all on the same page. Program designers Greg Tighe and Mike Rivera deserve congratulations for this breakthrough.

This word processor is easy to learn and is clearly aimed at the new Amiga owner. The manual is well organized and written, with clear screen diagrams, margin headings and a reasonably complete index. No one is credited with writing the manual, but whoever is responsible did a good job. One thoughtful touch is the space on page 1 for you to copy down the serial number for customer support reference. However, I would like to see more technical information included in the manaual for those of us who want it .

KindWords is not copy protected and there are easy-to-follow instructions for installing the program on a hard disk by dragging icons. If you don't have a hard drive, you'll really need two floppy drives or enough memory to use a RAM disk as an extra drive. One meg of memory is recommended — the program will run in 512K, but fonts, colors, graphics, etc., will be limited.

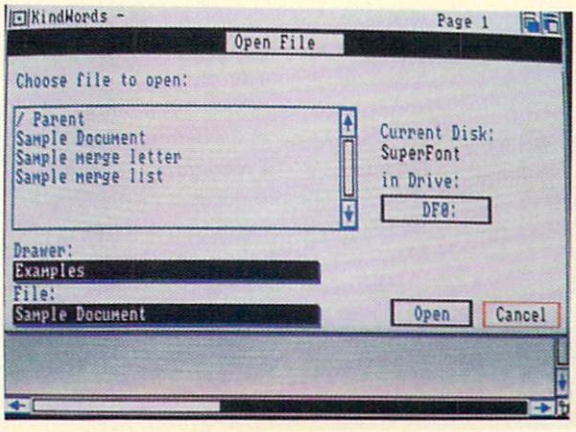
The screen

On screen, KindWords looks a lot like Excellence. It has the same kind of ruler with buttons for justification, spacing and tabs, including decimal. You can also make the ruler invisible. You can use KindWords's screen colors, which are the standard Preferences colors with black type on a white screen, or your own Preferences. Personally, I liked the KindWords selection.

One thing I didn't like about the screen was the way the screen buttons flicker in interlace mode. I like to use an interlaced screen for writing. You can see more of your work at one time, and it seems faster. However, this is a personal preference. I imagine most people won't use an interlaced screen with KindWords. (There is no mention of interlace in KindWords, by the way—I used the Preferences interlace option.)

(continued)

"...a word processor that gives great printouts, is designed for the average user and is offered at a good price."



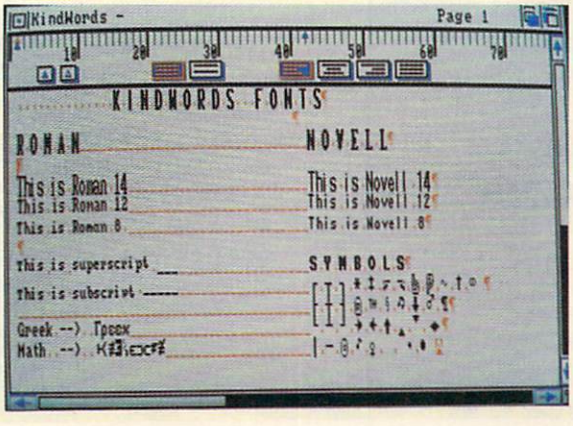


Figure one *(left)* The KindWords file requester occupies the entire screen.

Figure two *(right)* KindWords offers 3 text fonts, plus Math, Greek and Symbols.

Text entry

The program defaults to your Preferences margins. To fit everything on the screen, you need to set the right margin setting at a maximum of 76. If you set the margin wider, the program shifts one screen to the right and back again at the end of every line—slowly. A word processing tip: create a file with the defaults you use most often, with the FORMAT selection from the KindWords menu which lets you set margins, justification and tabs, all at once. Save this file on your data disk, then click on its icon to load KindWords and open the default file. When you start typing, remember to save it under a different name so it doesn't overwrite the original.

KindWords is easy to use but very slow in text entry. It's okay if you're doing straight typing, but deleting just one character in the middle of the text can pause the program for almost a second while it figures out the remainder of the paragraph and the page. This is a problem common to all graphics-based word processors, but KindWords seems to be unusually slow.

The program also has an odd trick of refreshing the screen from the cursor on down, then going back to do the top part. This can be confusing, especially since the mouse pointer gives no indication that there is work in progress. Several times I tried to sort out a screen full of broken or repeated sentences, thinking I had pasted text in the wrong place, only to see the screen suddenly right itself.

I also found myself repeating keystrokes after a long wait, thinking the first hadn't taken. (I know better, but I do it anyway!) But I've found a technique that can help speed up text editing in graphics-based word processors. If you have to do a lot of editing in the middle of a paragraph, position the paragraph at the bottom of the page. Then press RETURN right after the part you are editing , to separate the rest of the paragraph. That way, the program doesn't have to refresh the extra lines as you edit — just the part you are working on. When you are done, delete the RETURN to restore the single paragraph.

There are all kinds of ways to get around your KindWords document. You can move by character, by word, by paragraph, by line, by screen, all with ALT key combinations. Using the mouse can take forever, though, especially towards the end of a long document.

You can also select a word, a line, a sentence, a paragraph, a page, or the entire document with just the function keys. Nice. And much faster than dragging the mouse pointer to select.

Cutting and pasting is straightforward, as the program uses the system clipboard. You can also copy the ruler (i.e. format) from one part of the document to another.

Only one file can be opened at a time—a limitation in an Amiga word processor— but the program will prompt you to save your data before closing one file to open another.

KindWords' requesters are big—too big for my taste, especially since screen refreshing is slow. But I did like being able to cycle through directories by clicking on the disk button.

Advanced features

The "Advanced Features" section in the manual covers options like headers/footers, find and replace, the spell checker/thesaurus and print merge. Headers and footers appear in text windows that are separate from the main document. Each can be up to 15 lines long and include page number, date and time. Headers and footers can be excluded from the title page. A 100,000-word spelling checker, based on the Proximity/Merriam Webster Dictionary, is also included. When I first tried KindWords, I carefully copied the dictionary files to RAM: and made the appropriate ASSIGNments. But I didn't need to do all that work—KindWords will do it for you, if you have enough memory.

The spelling checker is slow but effective. I ran it on the sample document included, and it choked only on names and the spelling of "word processor" as one word. I felt a little offended on behalf of the document when the requester got to the end and announced "no more misspellings". A word is not necessarily misspelled because it is not recognized by a spell checker! The thesaurus is also very good. It lists a number of alternative meanings—each with the part of speech (noun, verb, etc.), the definition of the

word, and several synonyms. When I tried it out, I got six levels for the word "original", both noun and adjective, and four for "release", all verbs. "Release" as used in the text was a noun, however, so the thesaurus is not infallible.

Another improvement in KindWords 2.0 is automatic hyphenation. I was impressed with this feature—I rarely agree with the hyphenation choices made by word processors, but I checked all the hyphens in the first draft of this review against my dictionary and they all passed. In addition, the program offers a "soft" hyphen—you insert it manually and it shows up only if the word breaks at the end of the line. The manual offers a good tip. Warning that automatic hyphenation slows down typing, it suggests that you wait until you've finished the document, then turn on hyphenation to do the entire document.

A find -and -replace option will search for a combination of characters and spaces, replacing them with another combination using the same typestyle. You can't search for formatting codes like RETURN or UNDERLINE, but few word processors can. A "match case" option recognizes capital letters. A nice option is "whole word", which finds the string only when it is a complete word, not when it is part of a compound word.

Mail merge (called "print merge") is included and explained clearly and simply in the manual. You create a separate merge file either manually or by generating it with a database. The first record lists the field names, known in KindWords as "merge words". Records are separated by two RETURNs, fields by either a comma or a RETURN. You can include a comma in a field with CTRL+, and a RETURN with CTRL-RETURN.

When you create the main document, you include the merge words surrounded by angle brackets. The document prints out with everything in the right places.

Graphics

You can insert a graphic anywhere in your document. KindWords displays in medium resolution (640 x 200 pixels), and you can insert a low-res or medium-res image. KindWords will reduce a high-res image to medium-res by removing every other horizontal line of pixels. The program defaults to 4 colors to conserve memory and speed, but you can set it to display up to 16. The screen display doesn't affect printing; all the image's colors will be printed within the limits of the printer.

Graphics can be resized (proportionately or not) or cropped. If the graphic is too wide for the page, KindWords will crop it automatically. If it's too long, it will continue on the next page, but this will throw off the pagination.

KindWords cannot display or print text on the same line as a graphic. This means that the whole width of the page is allotted for the graphic. However, you can move the graphic from left to right in that space, positioning it anywhere you like.

Fonts

An important KindWords feature is its special high- resolution Superfonts. You can choose from Roman 8, 12 or 14, Novell, a sans-serif font that comes in the same three sizes, Superscript or Subscript (both Roman 8), Math, Greek and Symbols and—Hallelujah—bullets! In two sizes! Also, arrows, corners and pointing fingers.

There is a catch, though. These are the only fonts KindWords will recognize. If you don't have them in your system fonts directory (or boot with the KindWords disk), they will be listed on the menu but KindWords will politely but firmly ignore your attempts to use them. And it will not recognize other fonts at all. I tried several ways to outsmart it, but no dice. With this in mind, I would like to see The Disc Company add a larger headline Superfont —perhaps a 20 or 24 point size of Novell?

Printing

There are several printing options with KindWords. Draft quality gives a quick preview, using low resolution printer fonts. Final quality uses the printer's NLQ font in place of Roman 12 and KindWords' high-resolution Superfonts for the others. SuperFont quality prints the whole document in SuperFonts.

I tested KindWords with both my printers, a Panasonic 1091, several years old, and a PaintJet. With the obvious exception of color, the Panasonic won hands down.

PRINTERS THAT DO AND DON'T WORK WITH SUPERFONTS

The manual includes a list of KindWords' "superdrivers":

Okimate_20.PRN
ImagewriterII.PRN
HP_PaintJet.PRN
Okidata_293I.QUAD.PRN
EpsonX-No_NLQ.PRN
EpsonX.PRN
Okidata_293I.PRN
EpsonQ.PRN
CBM_MPS1000.PRN
Xerox_4020.PRN
HP_LaserJet.PRN

A note in the manual specifies that there are no special drivers at the moment for these Preferences printers:

Alphacom.Alphapro.101 Brother_HR-15XL Qume_LetterPro_20 Other printer limitations: The Readme file also mentions that non-graphics printers like the Diablo 630 and the Okidata 92 can't handle Superfonts. It's recommended that you use the HP LaserJet printer's own high-resolution fonts; a version of Superfonts for the LaserJet+ is available from The Disc Company. They are also updating the DeskJet printer driver. The Xerox 4020 has a hardware limitation with Superfonts; it's suggested that you stay with the printer's own fonts. The Okimate 20 has problems combining color graphics with text — the recommendation is to use a black ribbon and the black-and-white Preferences setting.

The Final Quality output on the Panasonic was great. I took the example file, which was in the default Roman 12, and scattered various examples of Superfonts throughout the text. In Final Quality, as promised, the Roman 12 was replaced by the printer's NLQ font; all other fonts were Superfonts. The program had no trouble switching back and forth between them, even for just a single word. On the Panasonic, the fonts looked good—clear and crisp, if a little denser than the NLQ font. This might be the result of using a new printer ribbon—since the printhead makes four passes in SuperFonts (vs. two for NLQ), it makes sense that a new ribbon might be too much of a good thing. The superscript and subscript, in Roman 8, were also clear and easy to read.

Superfont quality also gave good results. Roman 12 was not as nice a font as the printer's built-in NLQ, but the overall look was more consistent. Both modes showed graphics well.

The PaintJet was another story entirely. It was listed among the "superdrivers", and I looked forward to getting the same high-quality output. Instead, the Superfonts were sketchy and uneven—very disappointing. I tried printing in gray shades and black-and-white as well as color and experimenting with dip switches, but the fonts were the same in all of them. There is a "trouble shooting" section in the manual that offers fixes for common problems with specific printers, but there is no mention of the PaintJet.

With most printers, however, KindWords output looks great. Of course, there is a speed trade-off. When you first print out in Superfonts, you have to wait while the program checks for the Preferences printer selection, loads its own "Superdriver", if it has one, then loads in the Superfonts. (The second time is much faster.) Superfonts are printed with four passes of the printer, so they also take time. But they're worth the wait.

All around, KindWords is a nice piece of work —a word processor that gives great printouts, is designed for the average user and is offered at a good price ($99.00 list).

•AC•

A look at PageStream *desktop publisher from Soft-Logik*

by Barney Schwartz

Well, it was COMDEX in Atlanta (1988) and there were two really neat items shown for the Amiga. One was the LazerXpress. The other was a new desktop publisher from Soft-Logik which would accompany LazerXpress. This new DP program would be right up there with Professional Page and would do more since it could access the Virtual Page. This should be the program to move the Amiga into desktop publishing against the Mac and Big Blue.

Take heart, Amiganatics. Soft-Logik is shipping their desktop publisher. It is either in the mail to you or on its way to your favorite software supplier as you read this article. I have been working on Betas for over a year, and I have recently been writing the manual for the program. What I have discovered about the program follows.

The program should ship on two disks, but may expand to three very quickly because the program modules, fonts and utilities will grow to fulfill customer requirements. This program will let you write, format, design graphics, import and export text and graphics, spell check, auto-hyphenate, print to preferences or proprietary printers, treat text as a graphic object and much, much more.

Soft-Logik has worked very hard to make this program a stand-alone desktop publisher, able to handle any task required, from the first thought to final printed output.

The first release of PageStream is not fully functional. Due to a great deal of internal reorganization and dedicated work by certain key employees, there are no major bugs left in the program. There are an amazing number of features to cover in a program of this magnitude, and the sheer enormity of the task caused serious delays in production. All the basic features work, and more modules are quickly being completed. The first shipments include a coupon for a free upgrade. Now that is a class move that, to my knowledge, no other company has offered. But let me get into the program.

PageStream is a full-featured document processor. See the inset for a list of all features, all of which are available through pull-down menu options. Most of these options will open a requester window. For lack of an organized sequence in which to discuss the various functions, I will stick to menu order.

The Menus:

File menu

The file menu contains Append, Export Text, and Export Graphics—interesting items not found in programs of this type. The Append function lets you add to the working file from any source, whether it be another file, a picture, or text. You may append any or all of these at the current cursor position. Export Text and Graphics do just what their names imply—they let you export text or graphics files to any programs in your library. So you could ship a text file into WordPerfect or a graphic into DeluxePaint for later use or modification, and adjust the filenames accordingly from within PageStream without ever leaving your working document.

"...a full-featured, professional-level, document production tool."

Edit menu

The edit menu contains two unique items: Insert Mode and Replace Attributes. Insert Mode switches between insert and overwrite, with insert as the default. No great shakes, but it does provide a menu function as a switch so you don't have to remember to depress right <Amiga> <I> to toggle between the two.

Replace Attributes lets you search for font size, style, and page requirements, then replace them as you choose. This is indeed a timesaver when you must perform repetitive changes throughout a document. For instance, suppose your document was set by a typist with all titles underlined, and you want to change the font and italicize all titles. You simply search for underlined words and replace each with the new italic font.

(continued)

Layout menu

The unique items within the Layout menu are Insert Page, Delete Page and Move to Page. Why do I regard these seemingly standard functions as unique? Well, Insert Page lets you stick a page in the middle or beginning of your document. Try that with some of the other desktop publishers on the Amiga market and you end up with either an unsolvable crisis or a blown up document. After the insertion, PageStream will renumber your document.

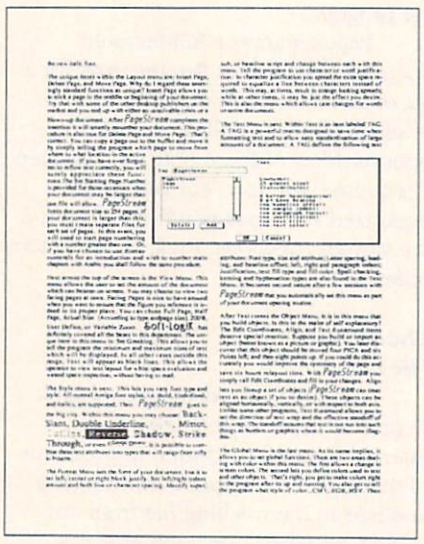
The same procedure holds true for delete page. You can copy a page out to the buffer and move it by simply telling the program which page to move in the active document. If you have ever forgotten to reflow text correctly, you will surely appreciate these functions.

The Set Starting Page Number is for those times when your document may be larger than one file will allow. PageStream limits document size to 254 pages. If your document is larger than this, you must create separate files for each set of pages. In this event you will need to start page numbering with a number greater than one. Or, if you use Roman numerals for an introduction and want to use Arabic for main chapters, you would follow the same procedure.

View menu

The view menu lets users set how much of the document can be seen onscreen. You may choose to view two facing pages at once. Facing Pages is nice when you want to ensure that the figure you reference is in its proper place. You can choose Full Page, Half Page, Actual Size (according to the type and page size), 200%, User Define or Variable Zoom. Soft-Logik has definitely covered all the bases in this department.

The unique item in this menu is Set Greeking, which lets you tell the program the minimum and maximum sizes of text which will be displayed. In all other cases outside this range, text will appear as black lines. This allows the operator to view text layout for white space evaluation and wasted space inspection without having to read.



Style menu

The style menu lets you vary font type and style, and supports all normal Amiga font styles (i.e., Bold, Underlined, and Italics). Then PageStream goes to the big city. You may choose Backslant, Double-Underlined, Light, Mirror, Outline, Reverse, Shadow, Strike-through, or even Upside-Down. And you can combine these text attributes into type ranging from nifty to bizarre.

Format menu

The Format menu sets the form of your document. Use it to set left, center, or right block justify. Set left/right indent amount and both line or character spacing. Identify super, sub, or baseline script, and change between each with this menu. Tell the program to use character or word justification. In character justification, you spread the extra space required to justify a line between characters instead of words. At times this may result in strange-looking spreads; at other times, it may be exactly just what you want . This menu also allows case changes for letters, words or entire documents.

Text menu

The text menu includes an item labeled TAG, a powerful macro designed to save time in formatting text, and to standardize large amounts of a document. A TAG defines the following text attributes: font type, size and attribute; letter spacing; leading; baseline offset; margin indent; paragraph format; justification; pattern fill and text color. The text menu also includes spell checking, kerning, and hyphenation types. After a few sessions, it becomes second nature to automatically set this menu as part of your document opening routine.

Object menu

The Object Menu is where you build objects. The Edit Coordinates, Align, and Text Runaround items deserve special mention. Suppose you build or import an object (better known as a picture or graphic), and discover later that this object should be moved four pica and six points left, and then eight points up. If you could do this accurately, you would improve the symmetry of the page and save six hours of layout time. With PageStream, simply call Edit Coordinates and fill in the changes.

Align lets you line up a set of objects (PageStream can treat text as an object). Objects can be aligned horizontally, vertically, or with respect to both axes. Unlike some other programs, Text Runaround lets you set the direction of text wrap and the effective standoff of this wrap. The standoff ensures that text is not run into borders or graphics, where it would become unreadable.

Global menu

The global menu lets you set global functions. Two areas in this menu deal with color. The first allows a change in screen colors. The second lets you define colors used in text and objects. That's right, you get to make colors right inside the program after it's up and running. You also get to tell the program what style of color (CMY, RGB or HSV). Then there are the Mechanical or Separable color types for overlay printing. If it comes from the rainbow spectrum, it is Separable; if not, it must be Mechanical (silver, gold, dayglow orange). You can edit the dictionary, set hyphenation rules, change kern pairs, determine the measurement system, tell the program which printer to use, and build macros to take care of all those redundant tasks.

Oh yes, I almost forgot the toolbox. This is a more or less the standard toolbox which sits where you tell it and lets you choose the tools to build your creations. The icons within are all self-explanatory. The toolbox can disappear or be recalled at a moment's notice whenever required.

Included items

The program comes in a sturdy slip-case. The manual is looseleaf, sectionalized and sized for easy use. The manual will take you through an introduction, then into tutorials to familiarize you with program operation. The heart of the manual is its reference section. Here you will find all the indepth operations of various menu functions and options available to the operator.

There is also an area which deals with overall publishing requirements, techniques and methods of operation, as well as a section of helpful hints, tricks and tips to make your life a little easier. As I have said so many times before, desktop publishing requires a lot of skill, artistic aptitude and creative ability. If you are lacking in any of these areas, you will face one of life's greatest challenges when you decide to publish on your own.

System requirements

Although you can run the program on a 512K Amiga with one disk drive, those of us who lived with small systems know this isn't any fun. It is highly recommended that you have, at bare minimum, one meg of RAM (the more the better), two floppies (a hard drive couldn't hurt), a laser printer for final output (dot matrix is OK, but, oh, so slow), and a graphics tablet if you are serious about doing your own art. You may think about getting a LazerXpress for fast output.

PageStream will talk to the virtual page, but the LazerXpress may not meet your reproduction requirements. Of course, if you are serious about being self-sufficient, you must eventually invest in a photo-typesetter to produce repro. Although small black-and-white jobs can be run on a laser, real work must be done on high-density negatives for color separations and/or long-run metal plates.

Program operation

The first thing to do after you boot your backup copy of the program is select all Global parameters. If you own only one printer, you will only need to set the printer configuration once. Next enter Set/Save paths and tell the program where to find the different directories you need. Set your desired measurement system before you open your document.

You may choose from any typesetting measurement systems including PICAS, DIDOTS, and CICEROS. The program will readjust things after the fact, but get in the habit of doing things correctly. This may save you a lot of anguish later. Besides, there are enough opportunities for error in a program of this scale.

You will eventually build a library of colors for objects using this menu. If you are concerned with correct spelling, you will spend some time adding or correcting words in the program dictionary. This will occur as you build files and recognize that not all the words in your vocabulary are in the dictionary supplied.

(continued)

Opening a file is a simple matter of choosing New or Open from the file menu. If you choose New you will be greeted by a window of standard Amiga gadgets on a grid representing a blank page. You should create a page format from scratch with create columns, or open your previously created Master Page. You must have a box open to begin text or graphics input.

Getting to work

I usually start by selecting Import Text. This brings up a requestor which asks for the file format of the job to be imported. You may choose ASCII, WordPerfect, or some as yet undetermined formats.

The requestor lets you bring in formatted or unformatted text. If you have ever been forced to realign all those sentences brought over from what you thought was a text-only save of WordPerfect, you will definitely appreciate this option. If you know you will be importing body copy, and you have a TAG set for this, you can save hours by importing this file after the TAG is selected. It is much simpler to go back and change attributes on the titles than to reset the entire body copy after an import.

By the way, when you are deciding on TAG parameters, PageStream is the only program which allows user modifications like x and y adjustment of font size, leading and kerning adjustments to 1/3600 of an inch, and 360-degree rotation of text (as individual letters, words or whole text areas). PageStream uses proprietary, scaleable fonts. This means no matter what size font you call (between 2 and 300 points), each character is smooth on the final output device.

It also means you are limited to PageStream fonts within your document. Some people will balk at this, but let's face fact—if you use a laser, you are limited by the fonts available in the printer. If you use a dot matrix, you are limited to the fonts available from the screen or printer ROM. In this case, you will get ten very good-looking fonts with the program, and Soft-Logik will provide a library of font disks for use as program sales expand.

Adding a graphic

Now it's time to import or build a graphic to support the text on this or a facing page. Yes, that's where pictures should be located—on the page or on a facing page. Have you ever read through a document that keeps referring to an illustration that's nowhere near the text? If so, you know exactly what I am talking about .

If you import graphics, first open a box in which to place the graphic. We do not have a clipboard to work in here, but it is nice to insert the graphic and watch its effect on the surrounding text. You select the size of the box and, if required, adjust positioning with edit coordinates.

Import Graphics will open a requestor which asks for information on file format. This is necessary because you may choose between IFF and other file formats. The program uses this information in its sizing algorithm (the routine which decides how to put the picture in the box you defined and still not lose any detail). That's right, PageStream will take almost any size picture, reduce or expand it, and put it in your box with all its original detail if possible.

Hopefully, while you were setting Edit Coordinates in the Object Menu, you also set Text Runaround. If not, the graphic covered part of your text and you must now fix it. You say you didn't import a picture? Instead, you are drawing your own from within the program. Good for you. To select an object, use the Toolbox (the thing with the miniature hammer and screwdriver).

Use the Object Menu to select Fill Style and color. Draw your object. Go back to the Object Menu if you must Align, Rotate, Lock, Unlock, Duplicate, or Reshape the object. If you have a group of objects you feel should be moved as a unit, use Group to coagulate them, or

Ungroup to dissolve them. Oh, I almost forgot! You can import pictures from other machines. Presently IMG and Pi3 files are supported.

When you are satisfied with this page, move on to the next. Keep this up until you have completed your document. Concentrate on getting the document together in a logical order. Try to ensure that graphics and text are coherent and aligned properly. If necessary, insert, delete, or move pages and objects.

You may find it advantageous to relink columns so important parts of various stories or articles begin toward the front of the document and continue within internal pages. You can do this with the column link (Set Text Routing) selection in the Layout Menu. It is much more efficient than creating new pages. Besides, you need a hook in the document for important items. PageStream is constructed as a professional tool for document makeup rather than a Hype piece of page makeup software.

Output devices

Now that you have composed your masterpiece, its time to do what you started out to do in the first place—print it.

You have a choice of any preferences-supported printer. You may also use a proprietary printer supplied with PageStream. Currently, the program supports Epson X or Q, C.ITOH, Toshiba, NEC, Xerox, Postscript, HP Laser and LazerXpress. This list is growing quickly.

The proprietary drivers provide much better gray scale and finer quality text than preferences could ever achieve. If you are doing color overlay plates, you should use the proprietary drivers to achieve the highest quality. If you need a driver, Soft-Logik will gladly build one for you.

Things to come

Some minor bugs have surfaced. Corrected versions of the program will be shipped to registered owners within a couple of weeks. Soft-Logik is redefining customer support and quick reaction to customer needs. It may be a little hard to get through on the phone, but have patience. If they cannot solve your problem immediately, they will get you an answer as soon as possible. They will not blow you off or leave you hanging.

Overall impression

PageStream is a full-featured, professional-level, document production tool. Although professional in application, it is extremely easy to use. Keep in mind, it is a complex program encompassing the entire gamut of publishing tools. It will take time to become proficient in all its facilities. This first release may introduce many new ideas and operating techniques which require some adjustment on the user's part. These techniques and facilities are standards within the publishing community and must be learned if you are serious about document composition. As Soft-Logik continues to add to the program, and provides competent customer support, they are assured a place of honor in the publishing arena.

Product Info

Soft-Logik Publishing Corp.
11131F South Towne Sq.
St. Louis, MO 63123
PH (314) 894-8608
BBS (314) 894-0057

PageStream $199.95

•AC•

Ray Tracing in AmigaBASIC

by Michael Morrison
AC Technical Editor

The process of ray tracing traces the path each beam of light travels, from source to object, from object to destination. Because of this, the picture produced has all the qualities of a real three dimensional object including, surface texture, shadows, colors, transparency and reflectivity, to name a few.

But anyone who has tried to produce a ray traced picture or who has seen one produced, realizes it can take an extremely long time to generate ray traced pictures. In the past, ray tracing programs were written in compiled or assembly languages to speed the process.

Abacus to the rescue. To stifle those who said it couldn't be done, Abacus has released another Amiga specific book, *Amiga 3D Graphic Programming in BASIC.* The book includes two programs, Editor and Tracer, that allow you to create ray traced pictures that will rival any other program available. And to top it all off, the programs are actually written in AmigaBASIC!

You are probably thinking that it can't be done, that the pictures generated are second rate. Not true. The pictures included on the demo disk we received are very nice. You're probably thinking it would take forever to ray trace a picture in a program written in AmigaBASIC. This time you are right and wrong. Some of the demo pictures took 3 weeks to finish! But, when generated with a compiled version of the program, the same picture took only 6 hours. That is comparable to other programs written in other compiled languages.

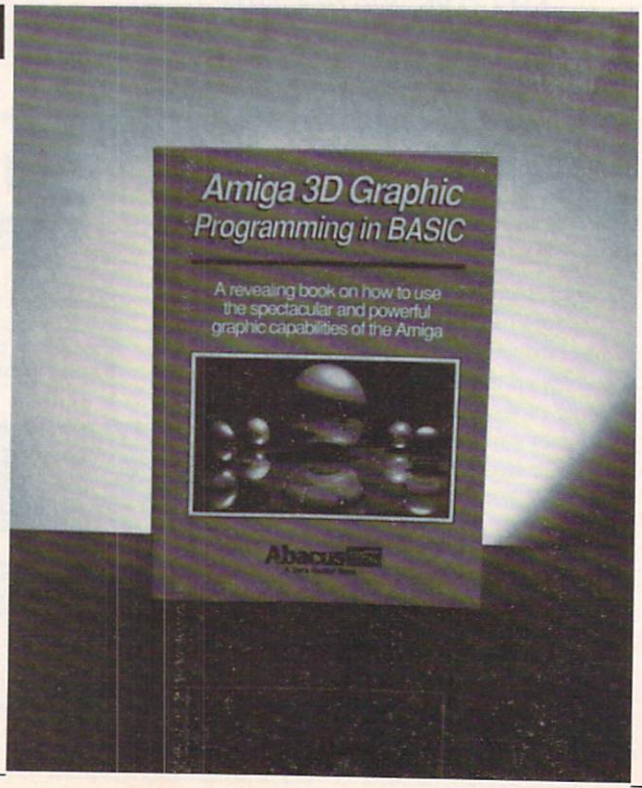
Amiga 3D Graphic Programming in BASIC contains a lot more than two BASIC programs. There is a complete chapter with indepth coverage of the basics of ray tracing. And don't let the thought of large mathematical formulas scare you. Although ray tracing uses a lot of math, the book breaks the formulas down to smaller, more easily digestible chunks.

Overall, it looks like Abacus has released another fine book that any semi-serious programmer will want on their book shelf. The Abacus book stands alone as the only book on ray tracing written in layman's terms.

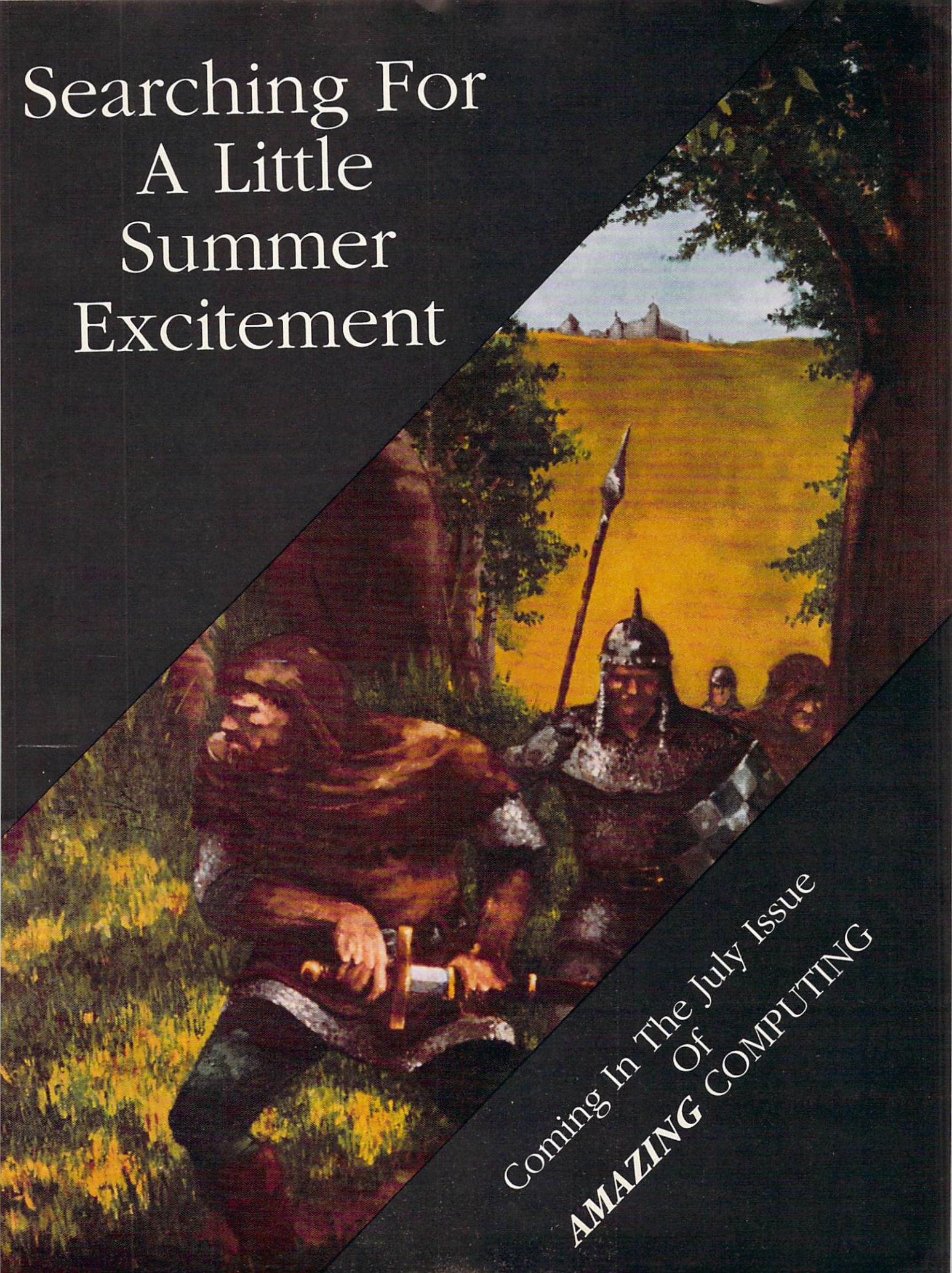
My only suggestion to anyone who buys the book is to spend the extra money and purchase the optional diskette. This will save you hours of typing and allow you to start ray tracing right away. The disk also has both the Editor program and Tracer program compiled under AC/BASIC V1.3 along with some previously defined objects that you can use.

•AC•

Amiga 3D Graphic Programming in BASIC contains a lot more than two BASIC programs. There is a complete chapter with indepth coverage of the basics of ray tracing.



Searching For A Little Summer Excitement



Coming In The July Issue Of **AMAZING** COMPUTING

PD Serendipity

Insight into the World of Freely Redistributable Software for the Amiga™

by C.W. Flatte

Fred Fish Disk 201

Draco
An update of Chris Gray's Draco. Improvements include support for floating point, register variables, code optimization, improved call/return standard, and more. This is version 1.2, an update to the version on disk #76. The documentation is on disk #77. By Chris Gray.

DropCloth
This program allows you to use a 2-bitplane (4-colored) IFF picture as the image on your Workbench. This is version 2.4 and it's shareware. By Eric Lavitsky.

Fred Fish Disk 202

SlavicFonts
A collection of Slavic fonts from Robin LaPasha. These are version 1.0, and some can be seen by clicking the two icons included.

Vlt
A VT100 emulator and a Tektronix emulator. The VT100 emulator is an enhanced version of Dave Weckeretal's VT100 emulator. The program has an ARexx port and supports XMODEM 1K/CRC and Kermit protocols. Version 3.656, binary only. By Willy Langeveld.

Fred Fish Disk 203

Examples
Assembly and C code examples, including some old favorites (like SpeechToy and YachtC3) rewritten in assembly language. Includes a replacement for the official audio device, R. J. Mical's file requester rewritten in assembler, a Type and Tell program that shows how to install a custom input handler ahead of intuition, and more. Authors: Jim Fiore and Jeff Glatt.

GurusGuide
Source files for all examples published in the "Guru's Guide, Meditation #1: Interrupts" by Carl Sassenrath, the architect of the Amiga's low-level multitasking operating system and designer of Exec. By Carl Sassenrath.

Isam
A library of routines to access relational database systems using the Index Sequential Access Method (ISAM). This is beta version 0.9. By Kai Oliver Ploog.

Fred Fish Disk 204

FileReq
Simple file requestor written in C. Must be run from the CLI. Source code included. By Jonathan Potter.

GnuGrep
The grep program from the GNU project. Replaces grep fgrep, egrep, and bmgrep. Version 1.3, includes source. Authors: Mike Haertel, James Woods, Arthur Olson, Richard Stallman, Doug Gwyn, Scott Anderson and Henry Spencer.

HAMCu
Installs a custom copper list for the current active view that shows all 4096 colors. A neat effect that shows off the color capabilities of the Amiga. Running the program again changes everything back to normal. Source code included. By Jonathan Potter.

Image-Ed
A shareware icon editor allowing you to design your own icons. The author suggests a shareware donation of $20 that will get you the source listing and registered as a user. Version 1.8, binary only. By Jonathan Potter.

JPCLOCK
A clock program loaded with features. Source included. By Jonathan Potter.

MouseBounce
A small hack that causes your mouse pointer to bounce around on the Workbench screen. It will continue bouncing until you close the MouseBounce window by clicking on the close gadget. Each click causes the pointer to move faster. Source included. By Jonathan Potter.

PopInfo
Puts an icon on the Workbench window that, when clicked, opens a window that gives you information about the status of your devices and memory. Allows you to display the bootblock and check for all currently known viruses. If found you can install that disk. This is version 2.9 and includes source. By Jonathan Potter.

PopDir
Similiar in appearance as PopInfo, but allows you to get a directory of all available volumes. Author: Jonathan Potter.

Teacher
You'll just have to run this one and see for yourself. Includes source. By Jonathan Potter.

Fred Fish Disk 205

Bally
Amiga port of the former arcade game named Click. This version is an update to the version released on disk #181. It has instructions written in German and is shareware. By Oliver Wagner.

BattleForce
A nicely done shareware game that simulates combat between two or more giant, robot-like machines. This is the latest version available, version 3.01. By Ralph Reed.

Chess
Update to the version first included on Fred Fish disk #96. It has been upgraded to use an Amiga Intuition interface. Has load/save, hint, switch sides, skip move, color palette and many more features. Version 2.0. Author: John Stanback; ported to Amiga by Bob Leivian. Version 2.0 upgrades by Alfred Kaufmann.

Fred Fish Disk 206

All this material is from the 1988 Badge Killer Demo Contest (BKDC).

Brownian
A demo based on both fractal theory and Brownian motion. Creates bizzare moving graphics picture. Includes source. By John M. Olsen.

Hawk
A stereo image of a hawk. Requires red/green stereo glasses for effect, but looks nice even without them. Author: Unknown (no documentation included).

MemFlick
This program treats your memory as a scrolling viewport. Interesting to look at. By Jim Webster.

PeX
Various demos strung together to show different aspects of the Amiga's graphics. If the source was included these would be a great tutorial. Nice job. The system crashed when I double-clicked on the Bugscript icon. Author: Unknown.

PictureGarden
A demo apparently done in compiled BASIC. Interesting slideshow. Author: Unknown (no documentation included).

StereoDemo
A demo of stereoscopic graphics, written in assembly language. Requires red/green stereo glasses to view. Includes sources. By David M. McKinstry.

Triple
Three demos showing some of the Amiga's graphics and sound capabilities. Unfortunately, only binary code included. By Tomas Rokicki.

Fred Fish Disk 207

Coyote
A nice animation with two cartoon characters that look a lot like Roadrunner and Coyote. Good job. The animation is large and distributed in "arc format", although I couldn't find arc on the disk. This is a 1988 BKDC entry by Gene Brawn.

Fred Fish Disk 208

AsteroidField
An animation of a spaceship weaving through a cluttered asteriod belt. This is a 1988 BKDC entry by Michael Powell.

Fred Fish Disk 209

Bowl
A Sculpt-Animate animation that shows three colored balls circling the rim of a mirrored bowl. Rendering the animation took about 2 weeks. Distributed in zoo format because of its size (zoo program included for easy unpacking). This is a 1988 BKDC entry by Vern Staats.

Dps
A program designed to work with the PrintScript program, a commercial PostScript interpreter for the Amiga, to provide a page previewer. Includes source. By Allen Norskog.

Fred Fish Disk 210

Calc
A very nicely done scientific/programmer/plotter calculator. The scientific portion has most of the operations found on the more popular handhelds. The programmer portion has number base conversions that allow conversions between hex, binary and decimal. The plotter portion will plot equations. Other features include 26 memories, full mouse or keyboard operation, pull-down menus, and iconization. By Jimmy Yang.

LabelPrint
A handy program that allows you to print labels for your disks. By Andreas Krebs.

NuHand
An animation of a large hand scraping its fingernails across an imaginary surface. Although the demo has sound, I was expecting it to be more like the sound that sends shivers down your spine when you run your fingernails down a chalkboard. This is a 1988 BKDC entry by Bryan Carey Gallivan.

The Amazing Computing Freely Redistributable Software Library announces the addition of...

New Orleans Commodore Klub's

inNOCKulation Disk

Version 1.5

To help inform Amiga users of the newer Amiga viruses and provide them with the means to detect and eradicate those pesky little critters!

Files and directories on the inNOCKulation Disk include:

Virus_Texts (dir)
Various text files from various places (Amicus #24, PeopleLink, and elsewhere!) describing the Virus(es) and people's experiences and their recommendations; TVSB "The Virus Strikes Back": satirical text describing future efforts to rid the universe of the dreaded (silicon) viruses! Interview with the alleged SCA virus author!

WB_VirusCheckers (dir)

VirusX3.2
Runs in the background and checks disks for viruses or non-standard boot blocks whenever they are inserted. (Recognizes several viruses and non-standard boot blocks. Removes virus in memory. Has a built-in "view boot blocks" & other features.)

Sentry
Revision of VirusX1.01 in Lattice C.

ViewBoot
Highly active mouse-driven disk and memory virus-checker which allows you to look at the pertinent areas (useful in case you supect a NEW virus!)

VRTest3.2
Watches memory for viruses; will alert the user and allow their removal if found. Can check & INSTALL disks, etc.

CLI_VirusCheckers (dir)

AntiVirusII
From The Software Brewery (W. German). Disables a virus in memory.

Clk_Doctor3
Corrects problems with the clock (caused by malignant programs, perhaps not really a "virus") (A500 & A2000)

Guardian1.1
Checks for attempts at viral infection at boot! Allows you to continue with a normal boot (if desired). Includes a small utility program to permanently place the program on a copy of your kickstart disk.

KillVirus
Removes (any?) virus from memory.

VCheck12:
Checks for SCA virus on disk or in memory.

VCheck19
Checks for any virus or otherwise non-standard boot block.

VirusKiller
A graphically appealing and user friendly program by TRISTAR.

Boot-Block_Stuff

SafeBoot2.2
SafeBoot will allow the user to save custom boot sectors of all your commercial disks and save them for such an emergency. If a virus somehow manages to trash the boot sectors of a commercial disk, just run SafeBoot and it will restore the boot sectors, therefore saving your disk!!

Virus_Alert V2.0.1
Yet another anti-virus program with a twist. Once installed on your boot disk a message is displayed just after a warm or cold boot notifying the the user that the disk and memory are virus-free, and forcing a mouse-button press before continuing.

BootBack1
Saves and restores boot-blocks. Runs from CLI only.

Antivirus *aka AVBB*
Includes SEKA assembler source.

XBoot
Converts a boot-block into an executable file, so you may use your favorite debugger (Wack, Dis, ...) to study it.

The inNOCKulation disk also includes icons and arc files.

To order the inNOCKulation disk, send: **$6.00** *includes postage & handling*
($7.00 for non-subscribers)

Amazing Computing
inNOCKulation disk orders
P.O. Box 869
Fall River, MA 02722

Roomers

by The Bandito

[The statements and projections presented in "Roomers" are rumours in the purest sense. The bits of infomation are gathered by a third party source from whispers inside the industry. At press time, they remain unconfirmed and are printed for entertainment value only. Accordingly, the staff and associates of Amazing Computing ™ cannot be held responsible for reports made in this column.]

Commodore watchers are still looking for Big C to move Amiga 500's into the mass market stores for the summertime. But it looks like Commodore isn't the only computer maker considering the mass market. IBM will be putting PS/2 Model 25's and 30's into Sears and Target stores. If this works, they plan to expand their mass-market penetration to other, as yet unnamed outlets. This step is designed to take away some clone sales and also to prepare the home market base for future IBM products.

Speaking of future IBM products, the top-secret computer the Bandito mentioned some time ago is still progressing. The Bandito has learned that it's now codenamed HPC, for Home Personal Computer. IBM has presented the specs to major software developers, particularly the entertainment software companies. There is plenty of interest—but skepticism—about IBM's ability to market computers to the home audience. (Memories of the PC*jr* and chiclet keyboards come to mind.)

According to those few who have seen it, the machine is technically impressive. The Bandito's informants speak of integrated DVI technology with a hot sound chip, MCGA graphics and animation coprocessing for somewhere under $2,000 to make a killer home/education machine. As you may recall, DVI (Digital Video Interactive) technology is a compression scheme for putting live video onto CD-ROM disks. Intel now owns the rights to DVI, and has announced their intent to make it into a one-board system by 1991 (and possibly a motherboard item on computers like the HPC). Of course, this doesn't mean that IBM will necessarily bring this technology to market in that form. They're probably just testing the waters, and what they finally decide to do is anybody's guess.

What would an HPC do to the Amiga market? That depends on the price point, and what Commodore does in response. The HPC could be a real threat if it was priced aggressively. It might be a good idea to get a CD-ROM player for the Amiga out soon, so when the price comes down there will be a market for Amiga CD-ROM's. Shouldn't be too difficult to get the Amiga to recognize the High Sierra format and keep track of emerging new standards. With the Amiga's sound and graphics capability added to the tremendous storage of a CD-ROM, you've got a killer combination. Let's hope Commodore's not asleep at the switch on this one.

Commodore R&D spending

Speaking of Commodore, many developers are starting to wonder just what is going on. What is their strategic vision for the future? How will they answer the challenges posed by Apple and IBM? These and many other questions will be hotly debated at the Developer's Conference in San Francisco this June. One of the things that has developers worried is Commodore's lack of commitment to research and development (R&D). For comparison, analysts usually look at R&D expenditures as a percentage of total revenue. For instance, IBM spent 10.2%, Sun 13.3%, and Apple a paltry 6.7%. What did multi-billion-dollar Commodore corporation spend? A staggering 1.7%.

The number looks even smaller when you consider that some of it was spent on C64 products and MS-DOS compatibles, Commodore's other two product lines. Small wonder that developers are concerned about Commodore's commitment to the Amiga's future. Will the situation change for the better? Perhaps DevCon will provide some answers.

The Bandito wonders if that R&D figure includes payments to Amiga developers working on key third-party software and hardware, or if that is tucked away in some other part of the budget.

The Amiga, once far ahead of other computer systems, is starting to fall behind. The introduction of the A2500 helps, but it's not enough. System throughput is critical, as is speed of memory chips and data storage devices. The Amiga, with its DMA channels (which are lacking on many so-called sophisticated computers like the Mac II) has a good start on being faster than its rivals. One of the most important features of the NeXT computer is its great system throughput, but the overhead with the system software means the advantage is mostly lost.

Part of the difficulty of creating newer, faster computers is that you have to speed up everything. Dropping in a 50 MHz 68030 won't help as much as it could unless you add faster memory chips, a 32-bit bus, etc. It's like dropping a jet engine into a Volkswagen—it'll go faster than normal, but to go all out you'll need better tires, shocks, brakes, a heavy-duty frame, and so on.

To be a real powerhouse, the Amiga 3000 needs not only a lot of hardware design, but system software that doesn't slow things down. That takes a great deal of work. With Motorola announcing the 68040, it seems to some that Commodore should maybe skip the 68030 Amiga and go straight to the '040. No way, says the Bandito. The 68030 is cheaper, and low cost should remain one of the Amiga's strong points.

Commodore hasn't even begun to address networking and connectivity issues. Sure, you can buy an expensive

(continued)

Ethernet card from a third party. But what about a network solution provided by Commodore? Something inexpensive like AppleTalk would be nice. And built-in file transfer software would be handy, too. Get with it, Commodore, or join Atari and the slide rule.

Other companies are pressing ahead with important technologies. A small firm called UVC Corp. has more advanced digital video compression schemes than DVI, further muddying the already murky waters of CD-ROM and all its variations. What's the latest score? Well, although Philips is still promising to bring CD-I to market in early 1990, it seems to be a dead duck as far as software developers are concerned.

Electronic Arts has reportedly cut back on its CD-I research efforts, and other developers are adopting a wait-and-see attitude before they spend more money on CD-I. They're ready to develop for some sort of CD-based system, but they want to wait until a standard emerges. The Bandito's guess is that things won't be sorted out until 1992 or so, but that there will be a standard to create a big market for software.

A new technology being developed at Oak Ridge National Laboratories uses a laser to polarize individual molecules on a disk for data storage. This leads to storage densities many orders of magnitude greater than those currently achieved. How does hundreds of trillions of bytes on a CD-ROM sound? That's terabytes of data. "Hey, where's my Library of Congress disk? I can't find it anywhere." Who cares if it's only write once—you can't live long enough to create that much data (well, OK, maybe if you use a video camera).

Even more new technology: ISDN (Integrated Systems Digital Network) is on its way, and it's going to mean big changes in the way we compute. Among other things, this new telecommunications standard uses standard phone lines for 144,000-baud data transmission, and on fiber optics it can do megabaud transmission rates. No more modems—just plug your telephone into your computer and send files about 60 or 120 times faster than you're used to.

Once the fiber optic-lines are everywhere, videophones will be easy to do. Maybe they'll make a videophone card whose output shows up in a window on your Amiga screen, which would be fun. Then people would sell software that makes your outgoing image look like a movie star, or lets you paint over an incoming call. What happens if you drag somebody's picture into the Trashcan? The possibilities are endless.

Apple II going bad?

The Bandito has already told you the Apple II line may cease production by the end of the year, according to some reports. Well, other sources are confirming this, and reports are even being published as news stories in trade journals. Of course, Apple stoutly denies this. When a computer is obsolete, you might as well sell as many as you can and let it die a natural death when retailers stop ordering it.

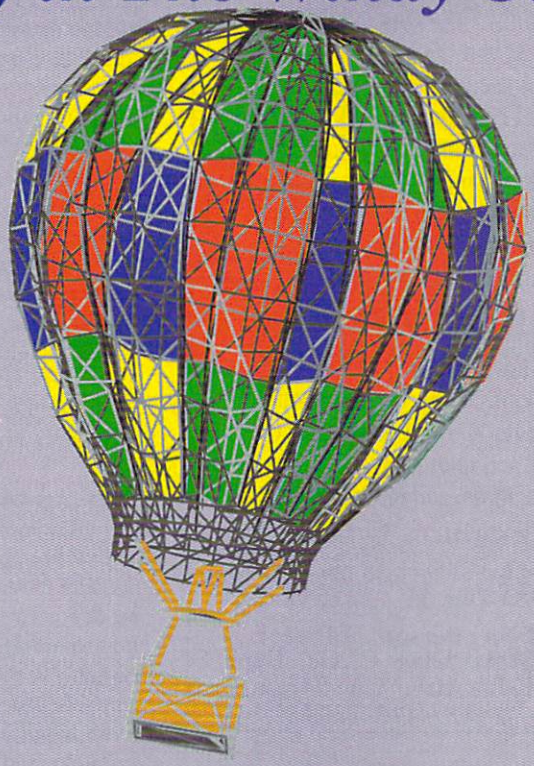
The Bandito figures Apple will keep selling members of the Apple II line for another year or maybe even two. Then one day somebody in the press will notice that nobody's bought an Apple II for months, and Apple stopped making them a while before. (Not unlike what the Bandito thinks will happen to the Commodore 64 in the same time period.)

The story goes that despite one or two loyal engineers, Apple is basically ignoring the Apple II products, and therefore the IIGS Plus may never see the light of day (besides, it would compete against the color Macintoshes). The pressure is growing for Apple to release a low-cost Macintosh, and at the same time developers are abandoning the Apple II market. Software sales for Apple II series computers are dropping faster than an Atari earnings report. If all you develop is Apple II software, then Apple doesn't really want to talk to you anymore. All their energies are directed at the Macintosh.

As part of the effort to allay the nasty rumors regarding the Apple II's health, Apple recently announced a genlock card for the Apple IIe and IIgs. Gee, what a great idea—wonder how they thought of it? It's interesting that Apple always refers to it as a "video overlay card" and never uses the word "genlock". The price? A mere $549. It's almost cheaper to buy an Amiga 500 and an AmiGen.

(continued)

COME ABOARD AmiEXPO
and We'll Blow You Away in The Windy City



AmiEXPO
The Amiga Personal Computer Show and Conference

July 28 - 30, 1989
The Chicago Hyatt Regency
151 East Wacker Drive
Chicago, Illinois

10,000 Attendees and 120 Amiga Companies Will Be There.
DON'T MISS THE FLIGHT!

Admission includes the Exhibition, Seminars, Keynotes & Amiga Artists Theatre!
120 Amiga Exhibitors Featuring State of the Art Software and Hardware, at the lowest prices!
Master Classes Available in Amiga Graphics, Video, Programming, Animation, Music and Publishing!
Seating for Master Classes is limited; call for schedule and availability before registering.
PRE-REGISTRATION DEADLINE IS JULY 14, 1989
For Hotel Reservations Call the Hyatt Regency at (312) 565-1234. Deadline for hotel reservations is June 26, 1989.
For discounted airfares, call American Airlines at (800) 433-1790 and give them this ID: S-83536.

Register by Mail, or Bring This Coupon to the show or Call 800-32-AMIGA Nationwide (or 212-867-4663)
For Your Ticket to The Amiga Event!

Yes, I want to come to AmiEXPO - Midwest

___ Friday ___ Saturday ___ Sunday

Registration is $5 Additional At The Door

One day - $15 _____
Two days - $20 _____
Three days - $25 _____

Master Class(es) - List Class and Time - $50 Each

_____ _____
_____ _____
_____ _____
_____ _____

Total Amount Enclosed _____

NAME _____
COMPANY _____
ADDRESS _____
_____
CITY _____ STATE _____ ZIP _____

For ___ MasterCard or ___VISA Payment
Expiration Date _____
Account Number _____
Name as it appears on card: _____
Signature _____

Make Check or Money Order Payable to:
AmiEXPO 211 E. 43rd St., Suite 301
New York, NY 10017

Bring Coupon to the door and get these Pre-Registration Prices!

Another sign of the demise of the Apple II line—the two leading Apple II magazines, *A+* and *InCider* (bitter rivals throughout their history), are merging. Where that leaves Bob Lindstrom, the Bandito isn't sure. Bob is/was the editor-in-chief of *A+*, which was exceedingly strange because Bob is secretly an Amiga fanatic.

The Bandito heard that Bob even kept an Amiga at the *A+* offices, where no doubt it made all those poor little IIgs's green with envy. Bob is also a hot composer who created the music for *Rocket Ranger*, among other titles. Maybe he'll cut an album of software music. Say, how about Cinemaware's Greatest Hits on CD? The Bandito, for one, would be happy to hear his favorite game music while working away. If you can't play the game, at least it'll sound like you're having fun.

Epyx recently cut loose some of the contractors doing various sorts of work for them—mostly programming. Looks like their creativity products division is taking the brunt of the cutback. Shadowy reports say there's no effect on the Skunk Works where the top-secret device is taking shape.

According to a few bytes of data leaking out of their network, most of the engineering problems of the TSD (top secret device) are under control—it's the marketing issues that are being troublesome. Like how much, and where's the software coming from, and when do we ship, and where do we distribute it?

The Atari-Nintendo legal battles have also muddied the waters for any new videogame entries. But if any product is to make a big impact at Christmas, it has to show up in force at the June CES show in Chicago. Other possible entrants into the Videogame Vendetta are also proceeding cautiously; each one seems to be waiting to see what the others will do. June CES promises to be *very* interesting.

Loose Lips Ship Chips

The new Agnus chip has shipped to dealers, according to Commodore. By the time you read this, the chips should be at local stores. The upgrade cost will be around $150 according to the Bandito's sources. It's good news for users, because now bigger bitmaps can fit in, larger and faster animations are possible, and multitasking will be more fun.

You have to feel sorry for developers, though, because they won't know just how many users have the new Agnus, and that makes it hard to figure out if your software should depend on it. Ah well, that's their problem. Let's just enjoy the sensation of endless CHIP RAM space. Of course, in a few months the power users will start agitating for a 2 megabyte Agnus. We power hungry types are never satisfied.

It's the slowest time of the year for computer hardware and software sales, which means those companies on the edge are in a very dangerous spot. The companies the Bandito referred to before are feeling even worse than they did, and no buyers have appeared to rescue either firm. One of the companies has reportedly lost its affiliated label distribution agreement with a large software publisher, and they must now scramble to put together their own distribution network.

Their payments to creditors are getting later all the time, and their new products may not be enough to pull their corporate chestnuts out of the fire. The other company the Bandito's been following is still on very shaky ground, and might not make the payroll this month. Both companies are still shopping for buyers or more capital, but no luck so far. The Bandito recalls a quote from a favorite science-fiction novel, "It sure is a bad moment when you decide to sell out. But a worse moment, the *worst* moment in the world is when you decide to sell out and nobody's buying." If they go, these companies won't be the first to die in the Amiga market, and they won't be the last.

For Christmas this year, the Bandito wants an Amiga 2500 with UNIX, an A2286 BridgeBoard running DOS and OS/2, an Amax emulator running Macintosh software and, of course, good old AmigaDOS. Talk about a software base! The only problem would be organizing your disk library and figuring out how to partition your hard disk. Of course, you'd want a really large monitor so you could locate all the windows that would be stacked six deep on the Workbench. Wow, you could have bugs in there that nobody's even dreamed of before. It's either a technophile's dream or his worst nightmare.

The Bandito is truly appalled at the incredible lack of play value in some of the games flooding the market. Some of these games have trouble loading properly, or the graphics are randomly smashed. The Bandito recently saw a game that doesn't even have a rulebook (part of the game must be figuring out how to play it). Many games don't even keep you busy for an hour.

No wonder there's piracy. What's the answer? The increasing shelf-space crunch at the retail stores should help weed out some of the weaker competitors, but the occasional dud from the major publishers will still find a home.

How do you find a good game and avoid the clunkers? Well, you can't always expect reviewers to give you a straight answer—after all, they want to keep getting free games. Get a recommendation from a friend or go see the game being played in a store.

It's best if you can try it out for yourself. And when you get a dud, write a scathing letter to the manufacturer and tell them exactly what you think is wrong with the game. That'll get them to take some notice. Mail a copy to your favorite magazine, too, and maybe they'll print it. "Stop them before they program again!"

•AC•

AC's TECH / AMIGA

For The Commodore

AMIGA : The saga continues

Diskless Compiling

Exploring ***AMIGA*** Disk Structures

Utilizing the 68881 math coprocessor Part II

UPS Units Part II

Using Requesters in AmigaBASIC

AMAZING REVIEW:
The CMI Processor Accelerator

At Your Request

Using Requesters in AmigaBASIC

by John Wiederhirn

Your friend gives you the latest copy of his new disk utility program designed to do everything under the sun. While using the program, you notice that all the requester boxes look basically the same: a couple of lines of text and two buttons, one says OK, the other says Cancel.

Still, it's more than you have done in your BASIC programs. Whenever you needed a simple answer from the user, you had to code one of those dreaded "instr$" loops to get a character, then use if...then statements to interpret its meaning. But you are not willing to learn C just for nice-looking requester boxes. So for now, you'll have to make do with "instr$" loops.

While working on your home budget program in BASIC—which took a couple of days just to get the proper look and feel for the user interface—you realize your programs would look much better if you could use requester boxes to get the answers to yes/no questions.

The good news is that now you can. The bad news is you're going to get your hands a little dirty, and you'll have to learn some new concepts. But the end result will be the ability to call up and use the "system-type" requester boxes in your own BASIC programs. These boxes look even nicer than the ones used in BASIC. You can do this little piece of BASIC magic by taking advantage of AmigaBASIC's ability to call any of the system routines by using the DECLARE FUNCTION and LIBRARY commands.

If you were writing in C, all you would need to do to use the system routines is set up the proper parameters, then do a simple function call. Since the operating system was really made to be accessed using C, setting up the parameters in C is quite easy.

For us, it's going to be a little harder. While calling the actual routine is about as easy, setting up the parameters is going to take a bit more work. C has things called "structures", and most Amiga system routines require that their parameters be given to them in structures. BASIC, on the other hand, has nothing really comparable to structures. It's not impossible, but we're going to have to understand structures a little more before we can "kludge" them together from BASIC.

Type in the program at the end of this article. When you're done, SAVE IT! I cannot stress enough how important it is to save this program before running it. Since we are fiddling with the operating system with these routines, a typo or other such error in the wrong place can and will crash the system, erasing everything in memory. Take it from me, some of those crashes are not only total, but rather spectacular as well. (Ever see a Guru with thick green vertical bars scattered on top of it?)

The program does not include any comments, and you may notice it is somewhat "modular" in construction. The various subprograms it contains were designed to be extracted and used in your own programs. Don't worry, every line of code execution is explained in detail below.

Before we get into the program, however, let me explain a little about structures, and the systems routines used in the program. If you already have a working knowledge of using structures from BASIC, and understand what the system routines AllocMem, FreeMem, and CopyMem do, feel free to skip ahead to the section on the program itself.

Structures and system routines

When writing in BASIC, you often need to keep linked together pieces of data which are not of the same type. An example of this would be an inventory program where each part you keep in stock has a seven-digit stock number, an eighty-character description, and a four-digit number telling how many you currently have in stock. In BASIC, if you had to keep track of this data for ten different parts, you would probably use ten-item arrays like this:

```
DIM StockNo&(10), Description$(10), HowMany%(10)
```

To get the information for any particular type of part, you would just get the data from the arrays for that item. To get the stock number of the part whose data is in array item 10, and put it into a temporary variable, you would use this BASIC code:

```
TempHowMany% = HowMany%(10)
```

Not pretty or elegant, but functional. However, this can become unwieldy quite rapidly as the number of data "fields" increases. BASIC just isn't very good at this kind of thing.

Along comes the C language. In C, there is something called a structure which allows different "fields" of data to be linked up under one main heading. Even better, you can make arrays of those structures. The set-up code for the previous data types in C is as follows:

```
struct PartData {
    long StockNo;
    char Description[80];
    int  HowMany; } Inventory[10];
```

Slightly longer in set-up, the code creates a structure identical to the BASIC code above. To get the same data as above (How-Many for #10), use the following code:

```
TempHowMany = Inventory(10).HowMany;
```

No more complicated, and no questions regarding what the HowMany%() array belongs to. Another advantage of structures is that they can have other structures inside them. But for reasons having to do with how C handles structures, they are usually handled using their addresses by using pointers; instead of holding structures, they use pointers to structures instead. (The function of a pointer is a bit too involved to get into here, so I will only explain them as they relate to the program.) One other advantage of elements of a C structure is that they are contiguous in memory, which is very important to how we can use them in BASIC.

A rather common structure used as a parameter in system routines is the IntuiText structure. Its C code looks like this:

```
struct IntuiText {
    byte FrontPen, BackPen;
    byte DrawMode;
    short LeftEdge;
    short TopEdge;
    struct TextAttr   *ITextFont;
    char *IText;
    struct IntuiText  *NextText; }
```

When one of these structures is created in memory, it is in the format descibed below:

Element	Offset	Size	Format
FrontPen	0	1	byte (0-255)
BackPen	1	1	byte
DrawMode	2	1	byte
LeftEdge	4	2	short integer (BASIC's %)
TopEdge	6	2	short integer
ITextFont	8	4	long integer (BASIC's &)
IText	12	4	long integer
NextText	16	4	long integer

The offset entry refers to where we must poke the value to get it in the right place, relative to the address of the structure in memory. The format tells what kind of "POKE" we should use to put the value there, and the size is the size of the element in bytes. The last three elements are not the data itself, but the data's address in memory. For this article, we will always set ITextFont to 0& (which tells the system to use the default font). The IText element points to a null-terminated (i.e., ending with CHR$(0)) string, and the NextText element holds either the address of the next IntuiText structure, or 0& which means there is no next structure (more on this later).

To set an area of memory up as an IntuiText structure from BASIC would require the following code, where ITAddr& is the base address of the structure:

```
POKE ITAddr&, FrontPen%
POKE ITAddr&+1, BackPen%
POKE ITAddr&+2, DrawMode%
POKEW ITAddr&+4, LeftEdge%
POKEW ITAddr&+6, TopEdge%
POKEL ITAddr&+8, 0&
POKEL ITAddr&+12, IText&
POKEL ITAddr&+16, NextText&
```

Remember those lines of code, because you will be seeing them in the program , in a slightly altered form.

Next comes the problem of where to put the IntuiText structure. In the Amiga, you can have more than one program running at once, and you need a way of telling other programs that memory you are using is "off-limits" to them. While the C language does this automatically for structures, BASIC does not recognize structures, so it has no such ability.

What we need is a routine which tells the other programs that we are using a block of memory, and to stay out of it. Fortunately, the Amiga's operating system provides just such a routine, AllocMem. Part of the Exec library, AllocMem can allocate memory for the sole use of the calling program. It is called from BASIC as follows:

```
Addr& = AllocMem&( Size&, Type& )
```

The Size& variable must hold the number of bytes desired, and the Type& variable for the sake of this program will always be 65537&. The address of the allocated memory is returned in Addr& or, if AllocMem cannot meet your needs, a value of 0& is returned.

Another important part of "polite" memory management on the Amiga is that once a program is done with memory it allocated, it should return it to the system using the FreeMem routine. If it does not return the memory, the available system memory will get "eaten away" because the system can never regain control of the allocated memory on its own without rebooting. In the most extreme cases, the system can crash because it simply runs out of memory due to such allocation. Therefore, it is very important to be sure FreeMem is used when you are done with an allocated block of memory. To call FreeMem, use the following BASIC code:

```
dum& = FreeMem&( Addr&, Size& )
```

Addr& is the address you received when the memory was first allocated with AllocMem. Size& is the number of bytes allocated. A "garbage" value is returned in dum&, which has no meaning but is necessary to use FreeMem from BASIC.

There is one more thing about FreeMem you should probably be aware of. FreeMem has a rather nasty habit of crashing the system if given either an address of memory not allocated by your program, or if the number of bytes you try to free is different from the number you originally allocated. If you are using this routine, make sure you have a saved copy of the program on disk, and try to avoid giving this routine incorrect data. One good way of making sure the data is correct is to make the call to FreeMem a remark, then put a PRINT statement after it showing the values of Addr& and Size&. Also, put such a

(continued)

PRINT statement after the call to AllocMem. When you run the program, you'll know the numbers for the allocated memory, as well as those for the memory you are trying to free. It may eat some memory, but as long as it doesn't get out of hand, it can be invaluable in debugging programs which use these routines. Then, when everything is fixed, delete the two PRINT statements, and remove the apostrophe (or REM) from the line for FreeMem.

The third system routine used in the program is called CopyMem. Rather straightforward, CopyMem is called from BASIC like this:

```
dum& = CopyMem&( source&, destination&, size& )
```

Again, dum& is a dummy variable needed to make CopyMem work properly from BASIC. Source& is the address the routine is to copy from, destination& represents where it is to copy to, and size& is the number of bytes to copy. The real advantage of using this routine is its speed. Since it is done in machine-language speed rather than BASIC, it is much faster than using BASIC loop.

These three routines are very useful and essential to using system-level resources on the Amiga. The program uses one more system routine, AutoRequest, which we will discuss in the next section. To use any system routines from BASIC, you must have the appropriate .bmap files in either the libs: directory of your WorkBench disk, or the current directory when the program is run. This program needs the intuition.bmap and exec.bmap files. If you don't have them, you can make them using the ConvertFD program from the BASIC Demos drawer on your AmigaBASIC disk. For more information on .bmap files, consult your AmigaBASIC manual.

The AutoRequester program

This program was designed as an example of how to create certain system structures and use certain routines. More specifically, the program uses the system routines AllocMem and CopyMem to set up some IntuiText system structures in memory, which are then used to call the AutoRequest routine, and are finally "de-allocated" using FreeMem.

The AllocMem, CopyMem, and FreeMem routines, as well as the IntuiText structure, were discussed above. The AutoRequest remains to be discussed. This routine is the workhorse of the program and, when used properly from BASIC, it allows the programmer to put up a "system" requester, and get one of two answers from the user, specified by the programmer. In this program, it is used to just put up a sample requester with the selections "OK" and "Cancel" for the user to choose. However, since the programmer sets what each button's text will be, they could say and mean almost anything (they could have just as easily said, for example, "Reindeer" and "Flotilla"). The AutoRequest function is called from BASIC as follows:

```
response% = AutoRequest%( window&, text&, LButton&, RButton&,
LFlags&, RFlags&, width&, height&)
```

Window& is the address of the "parent" window's structure. For the sake of this program, is what you get from WINDOW(7) in BASIC. Text& is the address of the IntuiText structure that defines the text in the body of the requester. It can be the first of a linked list of IntuiTexts (using the NextText field), which allows more than one line of text in the body of the requester (we will discuss how to link them below). LButton& and RButton& are the addresses of the IntuiTexts which hold the texts for the left and right button. LFlags& and RFlags& are special operating system (OS) flags and, for the sake of this program, we will set them both to 0&. Finally, Width& and Height& are the size in pixels of the entire requester.

Finding the proper settings for these is mostly guesswork, but setting Width to 47 + (8 * num. chars. in longest line) will work as long as the length, and the button text is not too long. Height is harder, but 50 works for three lines of text in the body plus the buttons. Depending on the button the user selects, the routine returns either a zero or one in response%. If the left is selected, it returns a one. Clicking the right returns a zero. Set up the parameters, call the routine, and check whether it returned a zero or one. That's all there is to it. The system does the rest.

Now let's move on to the program itself. Once it is typed in and saved, run it a few times. If it crashes, check for typos and try again. Try both selections. See how easy it is? And it looks a lot better than "Do you wish to continue...(y) or (n)". Once you're done playing with the program, list it, or read along in the back of the article, because we are now going to dissect its operation line by line. Let's begin.

```
CLS
ON BREAK GOSUB CleanExit
BREAK ON
PRINT " Press any key to start the demonstration..."
dum$ = INPUT$(1)
```

These first few lines clear the screen, set BASIC so that if the program is interrupted it can try to exit "cleanly", and then wait for the user to press a key before starting (an "old" way).

```
DECLARE FUNCTION AllocMem&() LIBRARY
DECLARE FUNCTION CopyMem&() LIBRARY
DECLARE FUNCTION AutoRequest%() LIBRARY
DECLARE FUNCTION FreeMem&() LIBRARY
LIBRARY "exec.library"
LIBRARY "intuition.library"
```

Here we tell BASIC the names of the functions we will be using, and where it can find them. See the end of the previous section for more on libraries.

```
Start:
Txt1$ = "Sample Autorequester Dialog —"
Txt2$ = "Up to 3 lines of 30 characters"
Txt3$ = "can be used in this dialog box"
LBttn$ = " OK "
RBttn$ = " Cancel "
Hand& = WINDOW(7)
resp% = 0
```

This sets up variables with the values we want to appear in the requester. Note that the variables must be specific data types.

```
DoReq Hand&, Txt1$, Txt2$, Txt3$, LBttn$, RBttn$, resp%
SUB DoReq (win&, Tx1$, Tx2$, Tx3$, lbut$, rbut$, r%) STATIC
```

The BASIC subprogram DoReq is the controller of all the other subprograms used to finally create a requester. The way it is written here allows three lines of thirty characters each for the body text, and a twelve-character string for each of the buttons. These values are changeable, depending on what you need in your programs, and you should understand how you change them by the time we are done here. Inside the subprogram win& is the address of the parent window, Txt1$ - Txt3$ are the three lines of body text, lbut$ and rbut$ are the left and right button text strings, and r% is the variable which will be used to pass back the user's selection.

```
r% = 0
wd& = 280&
ht& = 70&
Tx1$ = LEFT$(Tx1$,30)
Tx2$ = LEFT$(Tx2$,30)
Tx3$ = LEFT$(Tx3$,30)
lbut$ = LEFT$(lbut$,12)
rbut$ = LEFT$(rbut$,12)
```

These zero the response variable, set the variables for the width and height parameters of AutoRequest, and make sure the various text strings are not too long.

```
CALL CreateIText( 0, 1, 0, 7, 3, Tx1$, t1& )
CALL CreateIText( 0, 1, 0, 7, 13, Tx2$, t2& )
CALL CreateIText( 0, 1, 0, 7, 23, Tx3$, t3& )
SUB CreateIText( Pa%, Pb%, Md%, LE%, TE%, Txt$, Adr& ) STATIC
```

The CreateIText subprogram is pretty generic. It takes the values for all the elements of an IntuiText structure, then allocates space for one, and fills it with those values. Note the changing values for TE% between the three calls to it above. These three calls set up the body text of the requester, and all will be in the same "box" of reference, so they must be offset vertically to prevent overwriting each other. Pa% and Pb% are the codes for the color register to be used for the text and its background (sort of). They are integers here, but will automatically be converted to bytes by the POKE command. Md% is the drawing mode to be used.

Here all use the JAM1 mode (0), but they can have any value from zero to three. Try changing them to see what the other modes look like. It too is converted to a byte by the POKE routine. LE% and TE% refer to the offset in pixels from the upper-left corner of the imaginary box surrounding each text area (body, LButton, and RButton). We'll set the ITextFont to 0&, so CreateIText doesn't ask for it. The same is true of NextText (for now). Txt$ holds the string which is to be the text data for the structure. Finally, Adr& will be the variable CreateIText returns the address of the structure in.

```
Txt$ = Txt$ + CHR$(0)
Size& = LEN(Txt$)
memtype& = 2^0 + 2^16
```

Next, Txt$ is made null-terminating (by appending CHR$(0)), Size& is set to Txt$'s length in bytes, and memtype& is set (albeit cryptically) to 65537, as said before.


```
iAddr& = 0&
tAddr& = 0&
iAddr& = AllocMem&( 24&, memtype& )
tAddr& = AllocMem&( Size&, memtype& )
```

By setting iAddr& and tAddr& to 0& before calling AllocMem we ensure that, should there is a problem allocating either, its value will equal zero. We then allocate 24 bytes for the IntuiText structure itself, putting the address in iAddr&, and then allocate the number of bytes there are of string data, with the address going in tAddr&.

```
IF (iAddr& = 0 OR tAddr& = 0) THEN
    PRINT "Memory allocation problem!!!"
    STOP
END IF
```

This checks for any problems in memory allocation. If a problem occur, it would halt program execution. Save anything that needs to be saved. You should probably reboot as well.

```
dum& = CopyMem&( SADD(Txt$), tAddr&, Size& )
```

Quickly copy the string data from where BASIC is keeping it to the memory we have allocated for it. BASIC has a nasty tendency of moving the addresses of string data around at will, and there is no guarantee that if we used that address in the

(continued)

IntuiText, it would still point to the string. This way, we know where it is, and it cannot move on us.

```
POKE iAddr&, Pa%
POKE iAddr&+1, Pb%
POKE iAddr&+2, Md%
POKEW iAddr&+4, LE%
POKEW iAddr&+6, TE%
POKEL iAddr&+8, 0&
POKEL iAddr&+12, tAddr&
POKEL iAddr&+16, 0&
```

The various forms of the POKE statement here are used to set the elements of our IntuiText structure to the desired value. Different versions are needed because the system stores numbers in different formats, depending on what type they are. For more on POKE statements, see your AmigaBASIC manual.

```
Adr& = iAddr&
END SUB
```

Finally, the address of the set IntuiText structure is put into the variable which it will be passed back in, and the subprogram ends.

```
CALL LinkIText( t1&, t2& )
CALL LinkIText( t2&, t3& )
```

Back in DoReq, the LinkIText is called. The LinkIText subprogram pokes the second IntuiText's address into the NextText field of the first one. The first parameter is the address of the first IntuiText. The second is the second's address. Here we use it to link line two of the body text to line one, and to link line three to line two. This is how to get more than one line of text where the routine calls for only one IntuiText's address.

```
CALL CreateIText( 0, 1, 0, 5, 3, lbut$, lt& )
CALL CreateIText( 3, 1, 0, 5, 3, rbut$, rt& )
```

Creating more IntuiTexts, this time the ones for the left and right buttons' texts.

```
r% = AutoRequest%( win&, t1&, lt&, rt&, 0&, 0&, wd&, ht& )
```

Here we go. This passes the (hopefully correct) parameters to the AutoRequest routine and, with any luck, will get back the user's response in r%.

```
CALL FreeIText( t1& )
CALL FreeIText( t2& )
CALL FreeIText( t3& )
CALL FreeIText( lt& )
CALL FreeIText( rt& )
SUB FreeIText( iAddr& ) STATIC
```

Now, to be polite, we call FreeIText for every IntuiText we allocated. All we need to give the subprogram is the address of the IntuiText which we are "de-allocating". Remember, giving the wrong address to this routine will crash the system.

```
tAddr& = PEEKL( iAddr&+12 )
Size& = 0&
WHILE ( PEEK( tAddr& + Size& ) > 0 )
    Size& = Size& + 1
WEND
Size& = Size& + 1&
```

FirstFreeIText checks the IText field of the IntuiText to tofind the address of the related text memory that was allocated. Next, since there is no field which says how long the text data is, but we do know it's null-terminated, the subprogram scans the memory from the address of the text data until it hits a CHR$(0), and figures the number of bytes it scanned, which is the number of bytes allocated.

```
dum& = FreeMem&( iAddr&, 24& )
dum& = FreeMem&( tAddr&, Size& )
END SUB
```

The subprogram then calls the FreeMem routine to free the specified memory. If the addresses or sizes are wrong, it will also provide video fireworks, and a visit from the Guru. If everything is okay, the subprogram ends.

```
END SUB
```

That's it. DoReq is done, and returns control to the main program, passing back the user's response.

```
PRINT
PRINT "You clicked on the ";
IF resp% THEN PRINT "OK button." ELSE PRINT "Cancel button."
PRINT
PRINT "Try the demo again? -- (y) or (n)"
dum$ = INPUT$(1)
IF UCASE$(dum$) = "Y" GOTO Start
CleanExit:
LIBRARY CLOSE
END
RETURN
```

The main program then decodes whether the left or right button was pressed based on whether resp% is zero or one. It then prints the corresponding message. The program again uses the old method to determine whether the user wants to continue or end. If the user wants to keep going, the whole thing starts over from Start:. If the user decides to quit, the program closes the libraries it opened, and ends. The RETURN statement is just there to make the ON BREAK...GOSUB statement not cause an error. It doesn't actually do anything.

That is how you use requesters from BASIC. Not really all that complicated, but tricky in parts. The program's subprograms are designed to be easily portable to your own programs. Just save them by saving with the "A" option, and merge them in as needed.

Study the program until you understand how things relate. Then play with changing the different parameters of CreateIText, and perhaps even rewriting DoReq for more lines or longer buttons. BASIC doesn't have to be a second-rate programming language.

Listing One

```
CLS

ON BREAK GOSUB CleanExit
BREAK ON
PRINT "  Press any key to start the demonstration..."
dum$ = INPUT$(1)

DECLARE FUNCTION AllocMem&() LIBRARY
DECLARE FUNCTION CopyMem&() LIBRARY
DECLARE FUNCTION AutoRequest%() LIBRARY
DECLARE FUNCTION FreeMem&() LIBRARY

LIBRARY "exec.library"
LIBRARY "intuition.library"

Start:
Txt1$ = "Sample Autorequester Dialog -"
Txt2$ = "Up to 3 lines of 30 characters"
Txt3$ = "can be used in this dialog box"
LBttn$ = "   OK   "
RBttn$ = " Cancel "
Hand& = WINDOW(7)
resp% = 0

DoReq Hand&, Txt1$, Txt2$, Txt3$, LBttn$, RBttn$, resp%

PRINT
PRINT "You clicked on the ";
IF resp% THEN PRINT "OK button." ELSE PRINT "Cancel button."
PRINT

PRINT "Try the demo again? -- (y) or (n)"
dum$ = INPUT$(1)
IF UCASE$(dum$) = "Y" GOTO Start

CleanExit:
LIBRARY CLOSE
END
RETURN

SUB DoReq (win&, Tx1$, Tx2$, Tx3$, lbut$, rbut$, r%) STATIC

  r% = 0
  wd& = 280&
  ht& = 70&
  Tx1$ = LEFT$(Tx1$,30)
  Tx2$ = LEFT$(Tx2$,30)
  Tx3$ = LEFT$(Tx3$,30)
  lbut$ = LEFT$(lbut$,12)
  rbut$ = LEFT$(rbut$,12)

  CALL CreateIText( 0, 1, 0, 7, 3, Tx1$, t1& )
  CALL CreateIText( 0, 1, 0, 7, 13, Tx2$, t2& )
  CALL CreateIText( 0, 1, 0, 7, 23, Tx3$, t3& )
  CALL LinkIText( t1&, t2& )
  CALL LinkIText( t2&, t3& )

  CALL CreateIText( 0, 1, 0, 5, 3, lbut$, lt& )
  CALL CreateIText( 3, 1, 0, 5, 3, rbut$, rt& )

  r% = AutoRequest%( win&, t1&, lt&, rt&, 0&, 0&, wd&, ht& )

  CALL FreeIText( t1& )
  CALL FreeIText( t2& )
  CALL FreeIText( t3& )
  CALL FreeIText( lt& )
  CALL FreeIText( rt& )

END SUB

SUB CreateIText( Pa%, Pb%, Md%, LE%, TE%, Txt$, Adr& ) STATIC

  Txt$ = Txt$ + CHR$(0)
  Size& = LEN(Txt$)
  memtype& = 2^0 + 2^16

  iAddr& = 0&
  tAddr& = 0&
  iAddr& = AllocMem&( 24&, memtype& )
  tAddr& = AllocMem&( Size&, memtype& )

  IF (iAddr& = 0 OR tAddr& = 0) THEN
    PRINT "Memory allocation problem!!!"
    STOP
  END IF

  dum& = CopyMem&( SADD(Txt$), tAddr&, Size& )

  POKE  iAddr&,    Pa%
  POKE  iAddr&+1,  Pb%
  POKE  iAddr&+2,  Md%
  POKEW iAddr&+4,  LE%
  POKEW iAddr&+6,  TE%
  POKEL iAddr&+8,  0&
  POKEL iAddr&+12, tAddr&
  POKEL iAddr&+16, 0&

  Adr& = iAddr&

END SUB

SUB FreeIText( iAddr& ) STATIC
  tAddr& = PEEKL( iAddr&+12 )
  Size& = 0&
  WHILE ( PEEK( tAddr& + Size& ) > 0 )
    Size& = Size& + 1
  WEND
  Size& = Size& + 1&
  dum& = FreeMem&( iAddr&, 24& )
  dum& = FreeMem&( tAddr&, Size& )
END SUB

SUB LinkIText( IText1&, IText2& ) STATIC

  POKEL IText1& + 16, IText2&
END SUB
```

•AC•

A Summer Treat

This Summer is hotter than ever, thanks to AC!

Amazing Computing has lined up 3 HOT issues for the Summer months to provide a cool look at some special topics on the Commodore ***AMIGA***. Included in the regular Summer issues of AC will be the following special focus sections:

- A special ***AMIGA*** **Technical** supplement in June—This feature will provide Amiga "techies" and non-techs with some new programming insights.

- A special ***AMIGA*** **Games** supplement in July—The weather will be hot, but so will Amiga games. The games supplement will keep AC readers informed on the latest and greatest in Amiga games.

- A special ***AMIGA*** **Video** supplement in August—AC's video supplement will showcase the Amiga's powerful video abilities with articles to help users literally **see** what the Amiga can do.

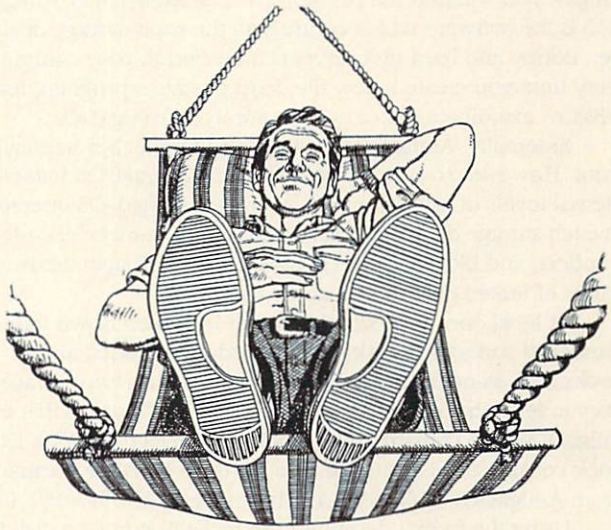
Amazing Computing always provides HOT information on the Commodore Amiga market no matter what the weather. But this Summer, the Heat is on as the Amiga continues to grow in the consumer market. AC will be there to guide its readers through the changes, developments, and new options the Amiga's rapid growth has provided.

Amazing COMPUTING™

Just the thing to ease those "Dog Days of Summer".

Relax

With a subscription to Amazing COMPUTING™



As the Summer heats up, don't miss an issue of AC's special coverage. Just think, AC is the perfect companion on a long vacation drive, a day at the beach, or just a warm afternoon in your favorite hammock.

Through the pages of AC, you will be able to keep current with the new Amiga trends and products. AC has provided more Amiga coverage than any other resource (Summer, Fall, Winter, and Spring). As an active Amiga user, this constant supply of Amiga information is important all year through.

Don't miss the Heat, subscribe today to keep current with the leader in Amiga information, Amazing Computing for the Commodore Amiga.

Don't Miss The HEAT!

Subscribe TODAY and save over 49% on Amazing Computing!

Amazing Computing is available to United States subscribers at $24.00 for 12 issues. That's a savings of $23.40 off the newsstand price. Canada and Mexico, $36.00. Foreign Surface to all other countries, $44.00.
(Air mail and first class rates available upon request.)

Yes! Please enter my 12 issue suscription to Amazing Computing today!

Name_____
Street_____
City_____ ST_____ Zip_____

MasterCard
VISA

Amount Enclosed_____
☐ Visa ☐ MC #_____ Signature _____

$20.00 minimum on all credit card orders.
All funds must be in U.S. currency drawn on a US bank. Please allow four to six weeks for processing.

1-800-345-3360

Exploring Amiga Disk Structures

by David W. Martin

AmigaDOS internals

As part of the ROM software that lies at the heart of most Amigas, you will find the computer's soul: AmigaDOS. AmigaDOS is the software which controls all the mass storage devices (i.e., floppy and hard disk drives) connected to your computer. Every time you create a new file, load or save a program, format a disk or examine a directory, you are using AmigaDOS.

Externally, AmigaDOS looks pretty simple, but internally it is not. However, to simplify, you can split AmigaDOS into two internal levels of operation. On level one, AmigaDOS operates on each storage device by breaking it into a number of surfaces, cylinders, and blocks. On level two, AmigaDOS operates on strings of linked disk blocks called files.

At level one, each storage device is broken down into a number of surfaces or disk sides, cylinders or tracks, and blocks. The standard AmigaDOS floppy disk has two surfaces, 80 cylinders, and 11 blocks per cylinder, which means that each AmigaDOS disk contains 1760 blocks (i.e., 80*11*2=1760). Each block consists of 512 bytes of data, so the total storage capacity for an AmigaDOS disk is 901,120 bytes (i.e., 1760*512=901,120).

Using the format described, AmigaDOS stores on disk files made from block-sized pieces of data and stored in a special format AmigaDOS can understand. Since there is a wide variety of disk hardware, AmigaDOS uses special software to drive it. By using a modular approach, AmigaDOS controls disk hardware connected to an Amiga computer. Level one of AmigaDOS consists of a collection of modules called device drivers for each device. The device drivers control the disk hardware by telling AmigaDOS how to read and write the disk blocks. You can add any new device to AmigaDOS by writing your own device driver or by using one supplied with the equipment (most commercial hard drives come with a prewritten device driver).

Now we move up to level two of AmigaDOS, which divides a disk into files and directories. At level two, AmigaDOS has software called file handlers. The file handlers are responsible for the format your files take when broken down to their smallest component—a disk block. The standard AmigaDOS file handler accepts commands that read, write, delete, or rename files. With a the file handler, the program does not have to understand the disk format used to store data. The program only has to ask for a file by name and wait for the handler to pass the data to the program. Since the program does not depend on the way the data is stored, you can have several file handlers that all use different formats. As long as the handler accepts standard commands, the rest of the system lets it manage that data in any way it wants. This is a definite advantage in a multitasking system like the Amiga, since special handlers can be written to support special applications while still retaining compatibility with the standard AmigaDOS file handler.

Disk structure

The current version of AmigaDOS (V1.3) supports two standard file handlers. The original file handler, which can be used on floppy or hard disks, and the new Fast Filing System (FFS). The FFS provides a way of greatly improving the performance of hard disks (and floppy disks with a special patch program). Both of these AmigaDOS handlers are compatible—the same set of commands manage files, and both communicate with standard device drivers to read and write disk blocks.

The only difference between the handlers is that they use different disk formats to store information on a diskette. The format used by the FFS is based on the original format, but the data block has been changed to provide better performance. Modifications to the data block when using the FFS remove duplicated data from the disk, speeding disk access and giving the disk more usable space.

A standard block of data contains 512 bytes of character data. AmigaDOS divides each block up into 128 words of data (a word is four bytes long and contains 32 bits). The first two blocks on an AmigaDOS disk, called boot blocks, are numbered zero and one. The blocks tell AmigaDOS what format is used. When the first word of the block zero contains the code DOS0, AmigaDOS knows the disk is formatted with the original file handler. A code of KICK tells AmigaDOS the disk is a Kickstart disk, and a code of NDOS means the disk uses a nonstandard format. The FFS uses the code DOS1.

When using either the original or fast filing system, you will find at the center of an AmigaDOS disk a block designated as the root block. The root block contains information about the disk's root directory, volume name, date the disk was first formatted, and some other important disk information. The format of an AmigaDOS root block is shown in Table One.

AmigaDOS root block

The first word of the root block, zero, indicates the type of block we are looking at. In this case, the root block type takes the value of T.SHORT, which is two. As you can see in Table One, AmigaDOS does not use words one, two, and four of the root block (word three will be explained below).

Word five contains the checksum for the root block. AmigaDOS uses this checksum to determine if the disk data has been corrupted. AmigaDOS calculates the checksum using the following method.

AmigaDOS first sets word five to zero and adds up all 128 words in the block. It then takes the sum of the 128 words and divides the sum by the maximum value that can be stored in a word, which is 4,294,967,296. The remainder becomes the checksum, and the checksum is placed in word five. Now when the sum is divided by the remainder of all the words, the answer should be zero. If it is not zero, something is wrong with the data block.

Word three contains the size of the root block's hash table, which is 72 for standard AmigaDOS disks. The hash table keeps track of the files in the root directory. When a new file is added to the directory, AmigaDOS takes the file name and performs a calculation on it. This calculaation is called the hashing function, and it returns a number between one and 72, which serves as an index into the hash table. The block number of the file's data is placed in the appropriate word in the hash table. Later, AmigaDOS uses the hashing function to find the block number of the file. If there is no file to match a selected hash value, then that entry in the table is set to zero.

The bitmap helps AmigaDOS decide which blocks are free and which contain data. Any time AmigaDOS adds or removes disk data, the disk's bitmap becomes invalid. Therefore, every time a file is deleted or created, the bitmap is marked as invalid before the actual bitmap is changed. When the appropriate blocks have been marked as free or used in the bitmap, the map is marked as valid again. The BMFLAG word (word 78 in the root block) indicates the bitmap's status. A value other than zero means the bitmap is valid. If the BMFLAG is zero, then AmigaDOS is in the process of updating the bitmap.

The bitmap is stored as one or more blocks containing the bitmap and a checksum. A standard AmigaDOS 3.5" floppy disk has 1758 blocks on it, so the bitmap contains 1758 bits. Since we can fit 8 bits in a byte (and four bytes in a word) each word contains the bit map for 32 blocks. Therefore, a standard AmigaDOS floppy disk uses 55 words for its bitmap (i.e. 55*32=1760 less two extra bits yields 1758).

The first word of a bitmap block, zero, contains a checksum for the block. The first bit of word one is the map bit for the first usable block on the disk. If this bit is set to one, then the first usable block is free. If the value is zero, then the block is used by some file or directory. The rest of the bitmap follows the same form. The second bit of the first word refers to the second disk block, and so on. If any of the slots in the bitmap table are not used, then AmigaDOS sets them to zero.

Following the bitmap table is a space for the current date and time. The three words (105-107) contain the days, minutes, and ticks (50 tics/second) since January 1, 1978. The format is called a date stamp, and it holds the day and the time the disk was last changed.

Following the date stamp is the volume name. It is stored in a format called BSTR or BCPL string (BCPL was the computer language used to write AmigaDOS). A BSTR string contains a number representing the string's length (0 to 255) and the string text. A disk volume named "AmazingCom" would have a length of ten, followed by ten bytes holding the word "AmazingCom." AmigaDOS volume names should be 30 characters or less in length.

Following the volume name is another date stamp in words 121-123. It holds the date and time the disk was formatted. The next three words (124-126) are not used in the root block, and the final word contains the block's subtype. The subtype value is ST.ROOT and its value is one.

(continued)

Table One Boot Block

Long Word Number	Byte Number	Description	Constant Value
0	0	Block Type: T.SHORT	2
1	4	Always Zero	0
2	8	Always Zero	0
3	12	Hash Table Size	72
4	16	Always Zero	0
5	20	Block Checksum	
6	24	Start of Hash Table	
:	:	:	
77	308	:	
78	312	BMFLAG (<>0, Bitmap is Valid)	
79	316	Bitmap Pointers	
:	:	:	
104	416	:	
105	420	Last Write Access: Day	
106	424	Last Write Access: Minutes	
107	428	Last Write Access: Ticks	
108	432	Disk Name (30 Character BSTR)	
:	:	:	
120	480	:	
121	484	Disk Creation Date: Day	
122	488	Disk Creation Date: Minutes	
123	492	Disk Creation Date: Ticks	
124	496	Always Zero	0
125	500	Always Zero	0
126	504	Always Zero	0
127	508	Secondary Block Type: ST.ROOT	1

Table Two User Directory Block

Long Word Number	Byte Number	Description	Constant Value
0	0	Block Type: T.SHORT	2
1	4	Block Pointer to Itself	
2	8	Always Zero	0
3	12	Always Zero	0
4	16	Always Zero	0
5	20	Block Checksum	
6	24	Start of Hash Table	
:	:	:	
77	308	:	
78	312	Not Used by AmigaDOS	0
:	:	:	
79	316	:	
80	320	Protection Status Bits	
81	324	Not Used by AmigaDOS	0
82	328	Directory Comment (79 Character BSTR)	
:	:	:	
104	416	:	
105	420	Creation Date: Day	
106	424	Creation Date: Minutes	
107	428	Creations Date: Ticks	
108	432	Directory Name (30 Character BSTR)	
:	:	:	
123	492	:	
124	496	Next Block Using Same Hash	
125	500	Pointer to Parent Directory	
126	504	Always Zero	0
127	508	Secondary Block Type: ST.USERDIR	2

Files and directories

AmigaDOS spends a great deal of its time managing disk files and directories. Most of the following disk blocks use formats similar to the root block. Table Two shows the format of a user directory block.

The first thing you will notice about the user directory block is its similarity to the root block. The root block is basically a directory itself, but it holds some special information about the disk, and it doesn't have a parent directory. The only differences you will find are in the first few words of the block. Word one becomes a pointer to the directory and tells AmigaDOS where the directory block is stored on the disk. The directory's hash table holds the file pointers for all the files in the directory using the same format as the root block.

At this point, the differences between the directory block and the root block become clear. For instance, areas used in the root block to store bitmap information are used differently in the directory block.

One way of using the space is to store an AmigaDOS file note for each file and subdirectory. The file note acts as a comment, which is stored as a BSTR, and tells you something about the file.

Another way of using the space is to store the file and directory protection bits. The bits, as listed in the AmigaDOS V1.3 documentation are Read, Write, Executable, Deletable, Script, Archive, and Pure. The protection information is stored in one word (32 bits). Since AmigaDOS uses only seven of the available 32 bits, there are 25 available for future expansion. Programmers should not tamper with the expansion bits since tampering with them may cause incompatibility problems with future versions of AmigaDOS.

The protection bits Read, Write, and Execute are not supported by AmigaDOS V1.3. A future release of AmigaDOS will prevent files from being examined, changed, or executed.

The Delete, Archive, Script, and Pure bits are supported by AmigaDOS V1.3. The Delete bit protects a file or directory from being deleted. Attempting to delete such a file produces an error message. The Archive bit tells AmigaDOS that a file has been changed since it was last accessed. The Script and Pure bits are newly supported protection bits in AmigaDOS 1.3. The Script bit marks files as script or batch files (i.e., the startup sequence). The Pure bit marks files that can be made resident in memory.

Following the protection bits is the comment field for the directory, the directory's creation date and time, and the directory's name. The directory's comment and name are stored in the BSTR format.

Following the directory's comment and name is a pointer to its parent or root directory. This pointer, contained in word 125, allows AmigaDOS to find its way back to the root directory from any subdirectory. This is accomplished by backtracking through the directory blocks using the pointer in word 125 as a step. Finally, the block subtype flag is present and it's called ST.USERDIR. The value of ST.USERDIR is two.

At first it appears that AmigaDOS directories would be limited to only 72 files per directory, but this limitation is overcome by the hashchain pointer (word 124). The pointer tells AmigaDOS where it can find the rest of the files that are

Table Three File Header Block

Lg Word Number	Byte Number	Description	Constant Value
0	0	Block Type: T.SHORT	2
1	4	Block Pointer to Itself	
2	8	Number of Blocks Stored in File Header	
3	12	Always Zero	0
4	16	First Data Block	
5	20	Block Checksum	
6	24	Last Block Pointer to Data Blocks	
:	:	:	
77	308	First Block Pointer to Data Blocks	
78	312	Not Used by AmigaDOS	0
:	:	:	
79	316	:	
80	320	Protection Status Bits	
81	324	Disk File Size in Bytes	
82	328	File Comment (79 Character BSTR)	
:	:	:	
104	416	:	
105	420	Creation Date: Day	
106	424	Creation Date: Minutes	
107	428	Creation Date: Ticks	
108	432	File Name (30 Character BSTR)	
:	:	:	
123	492	:	
124	496	Next Block Using Same Hash	
125	500	Block Pointer to Parent Directory	
126	504	Block Pointer to 1st Extension Block or 0	
127	508	Secondary Block Type: ST.File	-3

part of the current directory. Notice that if more than one file in a directory hash has a certain value, the hash table points to the first file with that value. Then that file points to the next file in the hashchain, and so on. If there are no more files in the hashchain, the file entry is zero. The hashchain allows an unlimited number of files in a directory limited only by the amount of disk space available.

File headers

The file header block is similar to the directory block in many ways. After you examine the directory structures above, you will know how to find a file in a disk directory. The following information will teach you AmigaDOS file structures.

The directory's hash table points to the file header blocks. Table Three shows the format of a file header. Once again, similarities arise, since the file header block is similar to that of the directory block.

File header blocks keep a list of the blocks used to store file data. A block table holds pointers to the file's data blocks (it takes the place of the directory block's hash table). The information is stored in the block table in reverse. For example, the first word in the table points to the file's last block, and the last word in the table points to the file's first data block. Zeros fill any of the unused slots in the list.

Most of the file header block is exactly like the directory block (protection bits, comments, time stamp, etc.) The only differences lie in the file size counter and the extension pointer. The file size counter holds the file size in bytes (word 81). The extension pointer (word 126) is used whenever a file exceeds 72 blocks in length. In a file larger than 72 blocks, AmigaDOS stores the remainder of the file's block information in a file list block. The extension pointer points to the file list block.

Although the file list block is similar to the file header block, it does not contain as much information. It simply holds pointers for the next 72 disk blocks and a pointer to the next file list block. The pointer to the next file list block is zero if no more blocks are used. Table Four shows the file list block's format.

A file data block contains a small header that describes the block and has a block type of T.DATA. The value of T.DATA is eight. The data block header consists of five words.

The first word of the data block points to the header block for that file. The second word contains the data blocks sequence in the file (a one for the first data block, a two for the second, etc.). The third word contains the number of data bytes

(continued)

Table Four File List Block

Lg Word Number	Byte Number	Description	Constant Value
0	0	Block Type: T.LIST	16
1	4	Block Pointer to Itself	
2	8	Number of Blocks Stored in File List	
3	12	Always Zero	0
4	16	First Data Block	
5	20	Block Checksum	
6	24	Last Block Pointer to Data Blocks	
:	:	:	
77	308	First Block Pointer to Data Blocks	
78	312	Not Used by AmigaDOS	0
:	:	:	
123	492	:	
124	496	Always Zero	0
125	500	Block Pointer to File Header Block	
126	504	Block Pointer to next Extension Block Zero if all Blocks are recorded in Data Block Table.	
127	508	Secondary Block Type: ST.File	-3

Table Five Data Block

Lg Word Number	Byte Number	Description	Constant Value
0	0	Block Type: T.DATA	8
1	4	File Header Block Pointer	
2	8	Data Block Number	
3	12	Number of Actual Data Bytes in this Block	
4	16	Next Data Block Pointer (0 if Last Block)	
5	20	Block Checksum	
6	24	Start of Data	
:	:	: (488 Bytes of Storage/Block)	
127	508	End of Data	

in the current data block. A block normally has 488 data bytes in it, but the last data block of the file may hold considerably less data. The fourth word contains a pointer to the next data block or zero if there are no more. The fifth word is the block's checksum. File data is stored from the sixth word onward.

See Table Five for a sample of a data block used by standard AmigaDOS. The Fast Filing System uses a specially formatted data block that contains 128 words of data. This is accomplished by dropping the data block checksum and other information in the data block header. By doing so, AmigaDOS is faster and provides more usable space on a diskette. Table Five shows the format of a disk data block.

Disk explorations

Exploring AmigaDOS disks is easy, but first you will need some sort of disk editor. Fortunately, a variety of commercial and Public Domain disk editors are available.

The best Public Domain disk editor is DiskX, written by Steve Tibbett of VirusX fame. The program is available from most local BBS and commercial networks and costs far less than its commercial competitors. The newest version even supports boot block backup, which allows you to copy your disk's boot blocks into a disk file. If the boot block is later damaged by a virus, DiskX will allow you to restore the boot block to its original form using the stored boot block data.

I recommend you get a copy of DiskX if you are interested in exploring AmigaDOS disks. Using DiskX's documentation and the information provided in this article, you are well on your way to understanding AmigaDOS disk formats and using your knowledge to restore or repair damaged and deleted files. DiskX does offer a feature for restoring damaged files, but this may not work in all cases. Since disk repair is difficult and depends on so many random factors, explaining it is beyond the scope of this article. But, simply put, the status of block pointers is the basis for restoring disk files. By learning how the block pointers work, you can learn how to repair disks. Hopefully, the knowledge presented above and tools like DiskX will speed you on your way. I have included a listing of some books and products to help you out. In the meantime, use this new knowledge and explore a few of your disks, but be sure to use backups until you are experienced with disk structures and disk editors.

Helpful books and programs

Books

AmigaDOS Manual, published by Bantam Books

Amiga Disk Drives Inside and Out, published by Abacus Software

Programs

DiskX—Public Domain software by Steve Tibbet. Disk editor with boot block save feature.

The Disk Mechanic—Commercial software published by Lake Forest Logic, Inc. Extensive disk-utility package featuring a disk editor, a disk-repair program, etc.

•AC•

The CMI Processor Accelerator

reviewed by Rich J. Grace

Many Amiga users interested in boosting the performance of their machines have invested in 32-bit accelerator cards. These usually run 68020/68881 chips with 32-bit memory. Such hardware usually works, but at a high price, and with slight software incompatibilities. CMI of Portland, Oregon has stepped forward with an alternative. It is a straight 68000 accelerator, running a 16 MHz 68000 chip at 14.3 MHz, with a socket for an optional 68881 floating-point chip driven by a clock crystal running at either 12.5 or 16 MHz. (The clocking for the 68881 can differ from and exceed the speed for the 68000 because the bus clock inputs for each chip are asynchronous. Hence, the math chip can run considerably faster than the CPU.)

The accelerator comes as a piggyback microprocessor-replacement board (for all machines, including the 2000) with the 68000 chip installed. Installation is fairly simple, requiring only a modicum of basic hardware experience. The documentation, bound in a palm-sized booklet, is functional but weak. It omits many details, such as a discussion of software packages expected to benefit from a math chip installation. Also omitted is a more extensive discussion of programming recommendations for users wishing to exploit the greater processor power now available.

A strange bird

As far as can be determined, there are no hardware incompatibilities with the CMI accelerator. This was thoroughly tested on the author's Amiga 1000 system with a previously installed "Insider" 1 Megabyte RAM piggyback board. The installation produced the bizarre configuration of two piggyback boards stacked on the microprocessor socket. In spite of this, the system booted seamlessly. This in itself spoke well for CMI's product-quality control and hardware compatibilities. Amiga 1000 owners with Insider boards can relax.

CMI also provides a set of "software switches" for installing the accelerator hardware into the system. The programs, InstallMC and ToggleSpeed (for installation of the 68881 and the 14 MHz 68000, respectively), are completely stand alone, and can be called from the Startup sequence or enabled from an icon. These programs are very efficient and small, and should fit on a crowded Workbench disk. One problem with the board's packaging is that no Benchmark programs are provided with these switch utilities (at least this was the case last September when the board was purchased). From the user's standpoint, this is not a wise omission.

Minimal speed increase

Overall system performance was not substantially affected by the CMI accelerator. The average spreadsheet calculation was about 20% faster. Graphics functions in programs such as DPaint and Deluxe Photo Lab ran anywhere between 10 to 30% faster. Benchmarks of programs such as the Sieve of Eratosthenes under Aztec C (see inset) showed no improvement whatsoever even with the 1.3 math libraries included in the compilation. Even windowing routines in Workbench showed little or no apparent enhancement in speed.

For any veteran Amiga user, the question is: "Why is the system performance so minimally affected?" Many users are no doubt aware that the Amiga runs a 14.32 MHz system clock. This 14.32 MHz bandwidth is divided equally between the Amiga graphics processor and the 68000 CPU. When a CPU accelerator is introduced into the Amiga system, possibly using frequency multiplier circuitry to boost the clock speed to the main processor, the "clock" driving the graphics chip remains unaffected. As a result, mundane operations, such as windowing and most conventional bitmapped graphics operations, do not benefit substantially from the faster CPU speed. (This is the case with the Amiga 2500.)

Light at the end of the tunnel

Some time after these tests were done, however, I obtained a disk of programs from a local vendor that were evidently tailored specifically to the accelerator. One program draws a sine wave pattern on the Amiga screen. The respective times for the three possible configurations (Plain vanilla 7 MHz, 14 MHz 68000, and 14 MHz 68000/16 MHz 68881) are as follows (in seconds):

Plain vanilla: 53.8
14 MHz 68000: 45.2
68000/68881: 2.2

The test showed a speed increase of 22 times over the original system. A second program draws a series of concentric circles on the screen. The comparative times for the same three configurations were as follows (in seconds):

Plain vanilla: 56.5
14 MHz 68000: 46.2
68000/68881: 8.6

The dramatic increase in speed reflected in the two programs compares exactly with the performance of the Amiga 2500 system.

This raises another question : "Do any ray-tracing programs or language compilers realize this level of performance, even when tailored to the 32-bit accelerators available?" Based on the tests I've has done, the answer is no. This is because the CMI accelerator architecture differs from other 32-bit cards in that it treats the 68881 as a peripheral, rather than as a true coprocessor. This is more than likely done out of necessity, as the 68881 was designed more as a coprocessor to the 68020 rather than to the less-advanced 68000. As a result, certain ray-tracing programs that support the 68881 (Sculpt 3D, for example) will crash on a CMI-installed machine.

(continued)

OOPS! Corrections

DeluxePaint III
In Louise Brinkman's Las Vegas COMDEX Report (AC Vol. 4.1, p. 48), some errors were reported regarding the list price and upgrade policy for **Electronic Arts' DeluxePaint III**. The actual list price for DeluxePaint III is $149.95. Also, Electronic Arts is offering two upgrade policies. The company will send DeluxePaint III to DeluxePaint II owners who send in the program's manual cover and $50.00 plus $7.00 shipping and handling. Any customer who purchased DeluxePaint II between December 1, 1988 and March 1, 1989 can upgrade to DeluxePaint III for $20.00 plus $7.00 shipping and handling. Customers must send original DeluxePaint II receipts and manual cover.

Printscript
In his AmiEXPO NY report (Vol. 4.4, p. 49), Steve Gillmor reported that C Ltd. manufactured the Postscript emulator **Printscript**. Printscript is, in fact, produced by **Pixelations**.

Spring '89 Product Guide

P. 40
Express Paint 3.0
Full-function paint program. Virtual pages, unlimited undo's, and 3-D perspective; rotating, distorting, stretching, and mirroring. Excels in test handling and printing. The only paint program that provides text fill into and around irregular shaped objects. Control font styles, sizes, justification, margins and line spacing to and from text. Supports PostScript, colored banners, and poster-sized output. Color cycling, 3-D anti-aliasing tools, gradient fills, more. $139.95 *Brown-Wagh Publishing (previously listed as Broderbund Software, Inc.)*

P. 98
Accolade *(address change)*
550 South Winchester Blvd. Ste 200
San Jose, CA 95128
(408) 985-1700

P. 104
Metran Technology *(address change)*
PO Box 82538
Tampa, FL 33628
(813) 978-3551

Our apologies to the companies involved and to our readers for any inconvenience this may have caused.

This raises a second question for the user: "Will software vendors begin supporting the CMI accelerator I have just purchased?" The answer to this is also a (qualified) no. I contacted several vendors to determine if any such plans were in the works. There were none. According to discussions with a representative from Seven Seas Software, (they put out DOUG'S MATH AQUARIUM—an excellent program which seems to be a natural for this type of hardware) one problem seems to be that CMI has not approached them about
support ventures. This really is a shame, given that other machines, such as the Macintosh, have very close-knit developer groups.

Conclusion

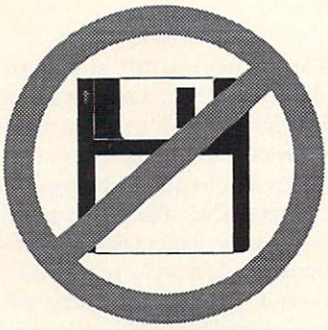
The CMI accelerator has great potential, particularly when the math chip is installed. In comparing operation times between the Amiga 2500 and a CMI-installed machine, the CMI did not suffer in certain applications. However, with only the 14 MHz 68000 chip installed, system performance is only slightly better. For an alternative to the prohibitively expensive 68020/030/881/882 accelerators to be effective, software support is an absolute necessity. However, as a programming tool, a CMI-installed system is very enticing.

Concerning compiler support, as noted in the Sieve Benchmark test, the compilation and run was done with Aztec C 3.6a. Admittedly, the author is not an expert programmer. Many readers may want to do some creative work to explore the 68881 configuration in the type of system discussed here. Those who do so and have interesting results to report can write to me in crare of AC.

Listing One

```
/* Sieve Benchmark
** Originally written in small C
** On the IBM at Byte
** Magazine. Adapted by Rich Grace.
** Recompiled under Aztec 3.6a.
*/
#include<stdio.h>
#define size 8190
#define LOOP 100
#define TRUE 1
#define FALSE 0

     int tblock[4];
     char flags[size + 1];

main()
{
   int i, prime, count, k, iter;
   puts("Sieve benchmark!\n");

   printf("%d iterations\n",LOOP);
      for(iter = 1; iter <= LOOP; iter++);
      {
      count = 0;
      for(i = 0;i <= size; i++)
         flags[i] = TRUE;
      for (i = 0;i <= size;i++)
         {
            if (flags[i])
            {
            prime=i+i+3;
              for(k = i + prime; k <= size; k += prime)
               flags[k] = FALSE;
            count++;
            }
          }
       }

}
```

•AC•



Diskless Compiling

Make development easy with **COMPILE**, a full-featured programmer's workbench.

by Chuck Raudonis

One of the biggest problems facing C programmers on the Amiga (besides trying to make sense out of the ROM Kernal Manual) is the long delay in compiling. We sit and watch the diskette drives grind and churn to load all of the include files, and the compiler creates all its intermediate files and proceeds with the compiles. The first and most obvious solution is to buy a hard disk drive, since the speed of the drive will solve the delays. Unfortunately, with the current pricing of hard disks for the Amiga, you would need a loan on par with the national debt to afford one.

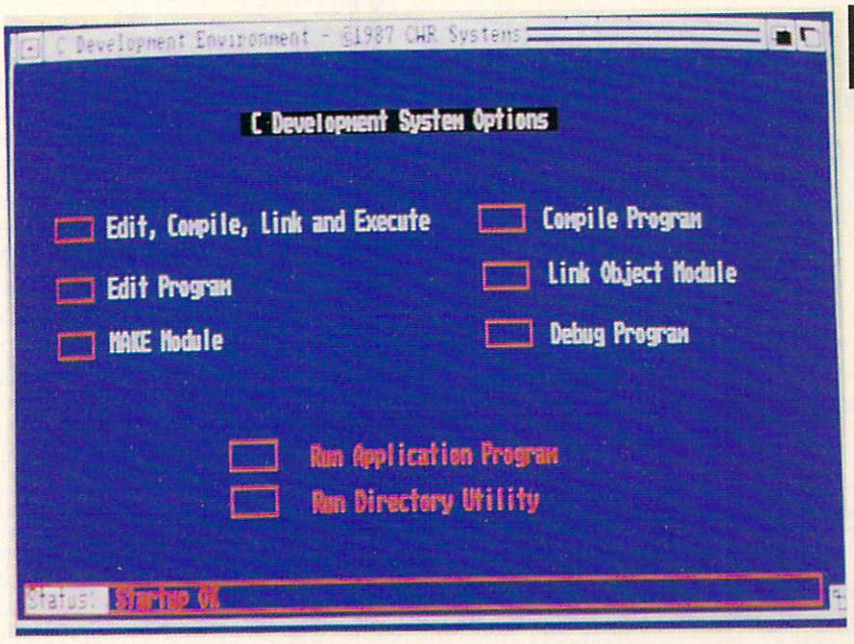


Figure One
The COMPILE Developer's Environment

Cheaper Solution

While the pricing of hard disks is high, the price of expanded memory is still relatively cheap. The price of the Michigan Software Insider board and the Spirit Technology boards are still within the reach of mere mortals. With a large bank of expanded memory, the required files can be transferred up into the RAM disk, and the entire process can proceed smoothly. You can tell the compiler to redirect the temporary files to the RAM disk, and the source module of the program to be compiled. Any required include files can be copied to the RAM disk. The entire compile can then proceed without disk access.

The only problem this solution poses is the large, complex amount of manual effort involved in identifying and copying all the requisite include files. The system include files are nested—include files load other include files. It is not uncommon to have 20 to 30 include files for an average program. This is a daunting task, to say the least. Even if one had the patience to load all these files by hand, the first time the GURU rears its ugly head, they are gone. The whole process must be then be started over.

A solution to the RAM disk problem is Workbench 1.3's recoverable RAM disk (RAD:). This device driver will allow a RAM disk to survive a visit to the GURU. This is not 100% foolproof because, if the program being developed really goes astray, RAD: can still be trashed, but the majority of crashes can be survived. Even with the benefits of the recoverable RAM disk, the programmer must still load the 20+ include files manually.

The COMPILE Developer's Environment

To solve this problem, I present "COMPILE", a developer's environment. COMPILE is a full-featured programmer's workbench. It is configurable and customizable. To augment COMPILE, I have presented a suggested startup-sequence and disk allocation to make development easy.

To use it, the programmer starts COMPILE and identifies the program to be worked on. COMPILE will then copy the program from the source diskette onto RAD:. While it is being copied, the program is parsed to locate any #include statements. These files are queued to be loaded to RAD: also. While the include files are being copied, they are parsed for nested includes. This process can go on indefinitely if you have includes within includes within includes.

COMPILE will look for the requested include files in the current directory. If the requested file is not in the current directory, COMPILE will search the include paths specified in the INCLUDES environment variable. If the required include file is already loaded onto RAD:, it will be detected before copying and will not have to be loaded again. Once all the required

(continued)

include files have been loaded, any include files currently on RAD: that the program doesn't need compiled are deleted to free up their space. This feature can be disabled by setting an option which causes COMPILE to retain all includes it loads onto RAD:. This will help speed things up if you switch between programs frequently. At the completion of the load step, the program to be modified has been loaded into RAD:, and all of the required includes are loaded with it.

COMPILE Gadgets

COMPILE has several gadgets and menus to control its actions. COMPILE has eight gadgets in its main window: Edit, Compile, Link, Make, Debug, Run Program, Run CLI utility, and one gadget labeled Edit/Compile/Link and Execute (See Figure One).

The first four gadgets are self explanatory as they Edit, Compile, Link, and Make the program being modified. The program is already loaded into RAD: so, if you use the suggested startup sequence, your favorite editor and the compiler and linker are also on RAD:. The process is very fast. A 40K source module will load into the editor in about 2 to 3 seconds. This includes the time needed to load the editor. If your makefile specifies additional object files to be linked to the main program, the makefile should specify these files as conditional for the compile and copy them to RAD: if they are not present. See the makefile supplied for making COMPILE for the syntax of this feature.

Totally RAD:

The Debug gadget will load Aztec's SDB™ and prepare the program for debugging. Since COMPILE assumes a total RAD: environment, SDB will look for the source, executable, and .dbg files on the RAD:. Currently, COMPILE does not copy these files off the disk. It assumes that frequent changes will be made during program development. Thus, the program must be run through a compile phase to get the executable and .dbg files onto RAD:. The Run Program gadget will execute the program you are working to test it. The Run CLI Utility will allow users to run their favorite mouse-based file manager (DIRUTIL, WIZARD, etc).to allow file manipulation without having to exit COMPILE.

The Power Gadget

The last gadget (Edit/Compile/Link) is the real power of the COMPILE environment. This gadget starts an intelligent compile sequence. Usually when a program is being developed, it is run through a series of Edit, Compile, Link, and test cycles. This gadget will automate the process. When this gadget is clicked, the program is loaded into your favorite editor, where you can modify it as needed. When the file is saved, and you exit the editor, the compile starts automatically. The output of the compiler is routed to a file on RAD: so any error messages are retained for future reference. If the compiler returns a non-error condition, COMPILE will then start the Link phase. The linker output is also directed to a file on RAD:. If the linker returns a non-error condition, the new version of the program is executed, and the fruits of your labor can be examined.

Back to the Editing Board

If an error condition is detected in either the compile or the link phase, the whole process is halted, and COMPILE restarts the editor so the problems can be resolved. If the editor you are using is a multi-buffer editor allowing more than one file to be loaded at once, both the error file and the source module are loaded simultaneously. You can toggle between the error messages and the source as you debug the program. If your editor does not support multi-buffers, then only the error file will be loaded. You must then make note of the errors, load the source file, and fix them.

Multi-Buffered Editing

COMPILE was really developed to make use of a multi-buffer editor. If your editor does not support multi-buffering, and you really don't want to change editors because you are accustomed to the command structure, take a look at an editor called UEdit, by Rick Stiles.

UEdit is a programmable editor. It comes with a predefined keyboard mapping, but the user can reconfigure it to emulate any editor or wordprocessor. Configuring the editor is easy, and it is amazing what can be performed in this wonderful program. This is one tool any serious programmer should look at. I use an editor I am very comfortable with on other machines, so I have configured UEdit to emulate that editor exactly. The keystrokes are identical, so I don't have to think about which machine I am on.

Once you've debugged the program, saved the file, and exited the editor, the whole compile, link, and execute process will start up once again. This allows an easy development cycle. Remember, since everything is on RAD:, the disk drives haven't

been accessed at all while this has been going on.
Since it is still possible to trash RAD: with a truly errant program, COMPILE has an option switch that will force it to copy the source, object, executable, and .dbg files back to the diskette each time the source module is changed. It is good to set this option early in program development. This will require some disk access, but it will allow recovery from a total system trashing with all of your code intact. When your program starts to mature and becomes relatively stable, this option can be turned off, and COMPILE will only copy the program back to the diskette when a new program is being loaded, or when COMPILE is exited.

The Main Menu Bar

There are four menu items on the main menu bar. The first is PROGRAM. This allows the user to specify the name of an existing program, or one to be developed. The QUIT option causes COMPILE to copy the source, object, executable, and .dbg files back to the diskette and delete all of the files it copied up to RAD: before it stops. This graceful exit leaves the system in the state it was in before COMPILE was started.
The second and third menu items allow configuration of COMPILE. Since COMPILE creates an environment file (.env) to go with each file it works on, the configuration can be set differently for each program. When a program is loaded into COMPILE, the corresponding environment file is loaded, and the switches and options for COMPILE are set as they were when the program was last worked on.

The second menu item lets the user set the various flags and switches that control the execution of the compiler, linker, and programs that COMPILE calls. The flags for the editor, compiler, linker, and the program being modified can be set here. When the specified program is executed, the flags specified in this menu are passed to the program. For example, when the compiler is called by COMPILE, it is called with the command:

```
cc -flags program.c
```

where cc is the name of your compiler. This, too, is customizable as we will later see. In addition to the flags, the libraries to be linked can be specified in this menu. If you are developing a program with only one or two additional object modules, and you really don't feel like making a special makefile, COMPILE will let you specify the names of additional object files that should be linked to this program to make the executable file. These files are not maintained by COMPILE and, as such, you are responsible for moving them to RAD:. If the files are small, you can leave them on the diskette and access them from there with minimal degradation. If the program is composed of many object files, a makefile is the best solution. But for quick little programs built from pieces of existing code, this is a nice feature.

If the program being developed is composed of many object files and is built via a makefile, there is a feature of COMPILE that eases this process. The last entry on the Flags menu allows the user to set the name of the program to be made. If the program consists of 5 different modules, each module should have the name of the master executable file specified here. No matter which of the five files is being edited at the moment, the make utility will be passed the name of the master program, and the executable file will be created correctly. If no program name is specified here, the name of the program currently being modified is passed to make.

The third menu item allows the customization of the "C" environment. It allows you to specify the program names for the editor, compiler, linker, and mouse-based CLI utility that are to be invoked when called for. In addition, the location of the intermediate files from the compiler can be redirected from an item on this menu. The default location is RAD: but, if for some reason you want to change this, you can from this menu.

The last item on this menu lets you save the current flags, switches, and options as the default environment. When a program is loaded into COMPILE for the first time, or a new program is started, it does not have a corresponding .env file. The defaults saved with this menu option will be used for these programs. Any changes made after loading the new program will then be saved in that program's .env file for future use.
The last menu item lets you set the options that control how COMPILE will handle the program being modified. Each item on this menu is a toggle menu gadget with a check mark to indicate the status of the switch. You can specify whether the program should be copied back to disk each time it is modified.

This is also where you specify whether the include files should be parsed when the program is loaded. If the program to be modified is an include file and will not be compiled, this option should be turned off so the whole parse is not completed. This will save time because RAD: doesn't need the other

(continued)

include files since the file will not be compiled. A switch can be set to indicate whether the editor being used has multi-buffer capability.

The last switch controls whether COMPILE deletes the include files left on RAD: from the last program loaded that are not necessary for the currently loaded program. With this switch on, RAD: space is conserved. However, if you are bouncing between several modules while you are developing, it would be faster to turn this option off and have COMPILE retain all the needed include files for all the modules that you are working on. This way, when the program is loaded into COMPILE, the include files are still loaded, and you are ready to go.

A couple of words about the source code and its contents. The file name requester which is called when a program is loaded is currently a string gadget in this version. If you have a copy of the source for Charlie Heath's file name requester and his safeclose function, the code is in the program to support it, but it is commented out (see the get_file_name() function). The makefile is designed to handle these two files. If you don't have a copy of the source, it is available in Public Domain. If you choose not to use the file requester, remove the references to the two files in the makefile and all will be well.

The Edit/Compile/Link/Execute gadget that invokes the "smart compile" feature of COMPILE will keep you in the loop until you get a clean compile. To break out of the loop without getting a clean compile, your editor must be able to return a non-zero return code. UEdit can be set to return a non-zero return when the "Abort" option is taken, and it will return a zero when you exit gracefully. You will have to experiment with your editor to see whether it can return a non-zero return if needed.

The suggested diskette configuration for COMPILE to work efficiently is as follows: Strip all unneeded files from the boot disk for your compiler. Your objective here is to get enough room to fit all of the include files on the boot disk. Take the compiler, assembler, debugger, linker, COMPILE, and any utilities you use heavily during the development process and put them on a second disk labelled C_Lib: in a directory called To_RamDisk. Remove all of the link libraries from the boot disk, and move the libs you use to the C_Lib: disk in a directory called Lib.

If you use the suggested startup sequence, these libraries and the utilities and programs will be copied up to RAD: where they can be accessed quickly. Adjust the size of RAD: according to the amount of memory in your Amiga and the number of programs you are copying to RAD:, as well as the type and size of the programs you usually develop. This balancing act might take a few tries to get right. I like to keep 100 to 200k free on RAD: after all the utilities and libraries are loaded.

I hope this will provide you with a user-friendly programming environment, as well as help make your development hours more productive. By no means is this the final or ultimate workbench, but it should give you a start. Take the source, and tweak it to your heart's content. Let me know about any improvements you are able to make.

Listing One
Suggested Startup-Sequence

```
stack 10240
mount rad:
fastfonts
if EXISTS rad:c
   echo "Vdisk already established"
   skip nocopy
endif
echo "Insert C_Lib disk in df1:"
md RAD:c
echo "Establishing VD0:C Directory"
copy C_lib:To_Ramdisk/#? RAD:c
path RAD:c
echo "Creating Needed directories"
md RAD:C_Compile
md RAD:lib
md RAD:s
md RAD:t
md RAD:make
md RAD:compile
md RAD:include
md RAD:include/exec
md RAD:include/intuition
md RAD:include/graphics
md RAD:include/devices
md RAD:include/libraries
echo "Establishing RAD:S Directory"
copy sys:s/Data! RAD:s
copy sys:s/Help! RAD:s
echo "Copying Link Libraries to RAD:"
copy C_lib:lib/ma32.lib RAD:lib/mx32.lib
copy C_lib:lib/c32.lib RAD:lib/
lab nocopy
echo "Setting System Paths and Assignments"
path RAD:c
path sys:bin
assign s: RAD:s
assign t: RAD:t
echo "Establishing Compile Paths"
set CLIB=RAD:lib/!C_lib:lib!SYS2:lib!!
INCLUDE=RAD:include!RAD:make!Manx_c:include  CCTEMP=RAD:C_Compile/
set FUNCLIST=SYS2:lint/manx.c DBINIT=s:.dbinit
echo "Insert C_Disk in df1:"
echo " "
cd C_Disk:source
run compile
Echo "ok"
```

Listing Two
Compile.h

```
#include "intuition/intuition.h"
#include "stdio.h"
#include "exec/types.h"
#include "exec/nodes.h"
#include "exec/lists.h"
#include "exec/libraries.h"
#include "exec/ports.h"
#include "exec/interrupts.h"
#include "exec/io.h"
#include "exec/memory.h"
#include "libraries/dos.h"
#include "libraries/dosextens.h"
#include "ctype.h"


#define INTUITION_REV 31L
#define GRAPHICS_REV  31L
#define STATUS_LINE &IText32
#define ALLOCSIZE 3000L
#define DELETEINCLUDES MenuItem4
#define MULTIBUFFER MenuItem3
#define PARSEINCLUDES MenuItem2
#define COPYDISK MenuItem1


struct Screen *Screen;
struct Window *Window, *Second_Window, *File_Window, *Main_Window;
struct IntuitionBase *IntuitionBase;
struct RastPort *rastport,*Second_rastport;
struct GfxBase       *GfxBase;
struct IntuiMessage *message;
struct Process *Process;

struct NewWindow FileNewWindow = {
     0,2,600,180,
     1,2,
     MENUPICK | NEWSIZE | REFRESHWINDOW | ACTIVEWINDOW |
     MOUSEBUTTONS | RAWKEY | MOUSEMOVE,
     WINDOWDRAG | WINDOWSIZING | SIMPLE_REFRESH | ACTIVATE | WINDOWDEPTH,
     NULL,NULL,"Choose File",
     NULL,
     NULL,
     40,45,0,0,
     CUSTOMSCREEN };

ULONG message_class;
APTR  message_address;
USHORT message_code;

char args[10][300];
char *argp[10];
int  i,exit_flag,rtncd,edit_load,header_ptr,dirptr,
     fileptr,include_path_count;
```

```
char
include_paths[10][50],header_files[50][50],junk[10][50],*strptr1,*strptr2,*strptr3,msg[80];
struct FileLock       *pdir;
struct FileInfoBlock  *dir_info;
static struct dirent *FirstEntry;
static struct dirent *NextEntry;

FILE *stdprn;

UBYTE
program_name[40],edit_flags[20],compile_flags[20],link_flags[20],link_libs[80];
UBYTE  runtime_flags[20],obj_to_link[50];
UBYTE  object_location[30],directory_utility_name[40];

UBYTE  edit_program_name[40],def_program_name[40],
       def_dir[30],compile_command[40],link_command[40];

UBYTE  temp[300],dir_files[100][50],dir_dirs[10][50],file_to_make[50];


/* Screens, Windows and Gadgets */


/* "Wait a minute screen and messages */

struct IntuiText WaitText4 = {
    3,0,JAM2,
    128,37,
    NULL,
    "Process",
    NULL
};

struct IntuiText WaitText3 = {
    1,2,JAM2,
    151,26,
    NULL,
    "",
    &WaitText4
};

struct IntuiText WaitText2 = {
    3,0,JAM2,
    52,26,
    NULL,
    "Running the",
    &WaitText3
};

struct IntuiText WaitText1 = {
    1,0,JAM2,
    87,14,
    NULL,
    "One Moment Please...",
    &WaitText2
};

#define WaitTextList WaitText1

struct NewWindow WaitWindowStructure = {
    41,19,
    365,54,
    0,1,
    NULL,
    WINDOWDRAG+WINDOWDEPTH,
    NULL,
    NULL,
    "Process Window",
    NULL,
    NULL,
    5,5,
    640,200,
    WBENCHSCREEN
};


/* String Requester with OK and Cancel Gadgets */

USHORT Requester_BorderVectors1[] = {0,0,57,0,57,14,0,14,0,0};
struct Border Requester_Border1 = {
       -2,-1,
       3,0,JAM1,
       5,
       Requester_BorderVectors1,
       NULL
};

struct IntuiText Requester_IText1 = {
       1,0,JAM2,
       3,2,
       NULL,
       "Cancel",
       NULL
};

struct Gadget Requester_Gadget3 = {
       NULL,
       227,51,
       54,13,
       GADGHCOMP,
       RELVERIFY+GADGIMMEDIATE,
       BOOLGADGET,
       (APTR)&Requester_Border1,
       NULL,
       &Requester_IText1,
       0,
       NULL,
       3,
       NULL
};

UBYTE SIBuffer2[80];
struct StringInfo Requester_GadgetSI2 = {
       SIBuffer2,
       NULL,
       0,
       79,
       0,
       0,0,0,0,0,
       0,
       NULL,
       NULL
};

USHORT Requester_BorderVectors2[] = {0,0,294,0,294,10,0,10,0,0};
struct Border Requester_Border2 = {
       -2,-2,
       3,0,JAM1,
       5,
       Requester_BorderVectors2,
       NULL
};

struct Gadget string_gadget = {
       &Requester_Gadget3,
       28,23,
       291,8,
       GADGHCOMP,
       RELVERIFY+GADGIMMEDIATE,
       STRGADGET,
       (APTR)&Requester_Border2,
       NULL,
       NULL,
       0,
       (APTR)&Requester_GadgetSI2,
       2,
       NULL
};

USHORT Requester_BorderVectors3[] = {0,0,57,0,57,14,0,14,0,0};
struct Border Requester_Border3 = {
       -2,-1,
       3,0,JAM1,
       5,
       Requester_BorderVectors3,
       NULL
};

struct IntuiText Requester_IText2 = {
       1,0,JAM2,
       18,3,
       NULL,
       "OK",
       NULL
};

struct Gadget Requester_Gadget1 = {
       &string_gadget,
       37,51,
```

(continued)

```
        54,13,
        GADGHCOMP,
        RELVERIFY+GADGIMMEDIATE,
        BOOLGADGET,
        (APTR)&Requester_Border3,
        NULL,
        &Requester_IText2,
        0,
        NULL,
        1,
        NULL
};


struct NewWindow RequesterWindowStructure = {
        8,16,
        346,78,
        0,1,
        GADGETDOWN,
        SIMPLE_REFRESH+ACTIVATE,
        &Requester_Gadget1,
        NULL,
        "Input your Choice",
        NULL,
        NULL,
        5,5,
        640,200,
        WBENCHSCREEN
};

/* Main Screen and control gadgets */

USHORT BorderVectors1[] = {0,0,33,0,33,8,0,8,0,0};
struct Border Border1 = {
    -2,-1,
    3,0,JAM1,
    5,
    BorderVectors1,
    NULL
};

struct IntuiText IText1 = {
    1,0,JAM2,
    46,0,
    NULL,
    "Debug Program",
    NULL
};

struct Gadget debug_gadget = {
    NULL,
    368,102,
    30,7,
    GADGHCOMP,
    RELVERIFY+GADGIMMEDIATE,
    BOOLGADGET,
    (APTR)&Border1,
    NULL,
    &IText1,
    0,
    NULL,
    13,
    NULL
};

USHORT BorderVectors2[] = {0,0,607,0,607,9,0,9,0,0};
struct Border Border2 = {
    -2,-1,
    3,0,JAM1,
    5,
    BorderVectors2,
    NULL
};

struct Gadget Gadget12 = {
    &debug_gadget,
    11,186,
    604,8,
    GADGHCOMP,
    RELVERIFY,
    BOOLGADGET,
    (APTR)&Border2,
    NULL,
    NULL,
    0,
    NULL,
    12,
    NULL
};

USHORT BorderVectors3[] = {0,0,34,0,34,9,0,9,0,0};
struct Border Border3 = {
    -2,-1,
    3,0,JAM1,
    5,
    BorderVectors3,
    NULL
};

struct Gadget directory_utility_gadget = {
    &Gadget12,
    174,155,
    31,8,
    GADGHCOMP,
    GADGIMMEDIATE,
    BOOLGADGET,
    (APTR)&Border3,
    NULL,
    NULL,
    0,
    NULL,
    11,
    NULL
};

USHORT BorderVectors4[] = {0,0,35,0,35,9,0,9,0,0};
struct Border Border4 = {
    -2,-1,
    3,0,JAM1,
    5,
    BorderVectors4,
    NULL
};

struct Gadget run_gadget = {
    &directory_utility_gadget,
    173,140,
    32,8,
    GADGHCOMP,
    RELVERIFY+GADGIMMEDIATE,
    BOOLGADGET,
    (APTR)&Border4,
    NULL,
    NULL,
    0,
    NULL,
    10,
    NULL
};

USHORT BorderVectors5[] = {0,0,27,0,27,8,0,8,0,0};
struct Border Border5 = {
    -2,-1,
    3,0,JAM1,
    5,
    BorderVectors5,
    NULL
};

struct Gadget make_gadget = {
    &run_gadget,
    40,103,
    24,7,
    GADGHCOMP,
    RELVERIFY+GADGIMMEDIATE,
    BOOLGADGET,
    (APTR)&Border5,
    NULL,
    NULL,
    0,
    NULL,
    9,
    NULL
};
```

```
USHORT BorderVectors6[] =
{0,0,33,0,33,8,0,8,0,0};
struct Border Border6 = {
    -2,-1,
    3,0,JAM1,
    5,
    BorderVectors6,
    NULL
};

struct Gadget link_gadget = {
    &make_gadget,
    367,83,
    30,7,
    GADGHCOMP,
    RELVERIFY+GADGIMMEDIATE,
    BOOLGADGET,
    (APTR)&Border6,
    NULL,
    NULL,
    0,
    NULL,
    8,
    NULL
};

USHORT BorderVectors7[] =
{0,0,33,0,33,8,0,8,0,0};
struct Border Border7 = {
    -2,-1,
    3,0,JAM1,
    5,
    BorderVectors7,
    NULL
};

struct Gadget object_gadget = {
    &link_gadget,
    365,64,
    30,7,
    GADGHCOMP,
    RELVERIFY+GADGIMMEDIATE,
    BOOLGADGET,
    (APTR)&Border7,
    NULL,
    NULL,
    0,
    NULL,
    7,
    NULL
};

USHORT BorderVectors8[] =
{0,0,28,0,28,7,0,7,0,0};
struct Border Border8 = {
    -2,-1,
    3,0,JAM1,
    5,
    BorderVectors8,
    NULL
};

struct Gadget edit_gadget = {
    &object_gadget,
    39,85,
    25,6,
    GADGHCOMP,
    RELVERIFY+GADGIMMEDIATE,
    BOOLGADGET,
    (APTR)&Border8,
    NULL,
    NULL,
    0,
    NULL,
    6,
    NULL
};

USHORT BorderVectors9[] =
{0,0,29,0,29,7,0,7,0,0};
struct Border Border9 = {
    -2,-1,
    3,0,JAM1,
    5,
    BorderVectors9,
    NULL
};

struct Gadget full_process_gadget = {
    &edit_gadget,
    38,65,
    26,6,
    GADGHCOMP,
    RELVERIFY+GADGIMMEDIATE,
    BOOLGADGET,
    (APTR)&Border9,
    NULL,
    NULL,
    0,
    NULL,
    5,
    NULL
};

/* Gadget list */

struct IntuiText IText2 = {
    3,1,COMPLEMENT,
    20,1,
    NULL,
    "Delete Uncalled Includes",
    NULL
};

struct MenuItem MenuItem4 = {
    NULL,
    0,33,
    332,10,
    CHECKIT+ITEMTEXT+COMMSEQ+MENUTOGGLE+
      ITEMENABLED+HIGHCOMP+CHECKED,
    0,
    (APTR)&IText2,
    NULL,
    'D',
    NULL,
    0xFFFF
};

struct IntuiText IText3 = {
    3,1,COMPLEMENT,
    20,1,
    NULL,
    "Multi-Buffer Editor",
    NULL
};

struct MenuItem MenuItem3 = {
    &MenuItem4,
    0,22,
    332,10,
    CHECKIT+ITEMTEXT+MENUTOGGLE+ITEMENABLED+
      HIGHCOMP+CHECKED,
    0,
    (APTR)&IText3,
    NULL,
    NULL,
    NULL,
    0xFFFF
};

struct IntuiText IText4 = {
    3,1,COMPLEMENT,
    20,1,
    NULL,
    "Parse #include files",
    NULL
};

struct MenuItem MenuItem2 = {
    &MenuItem3,
    0,11,
    332,10,
    CHECKIT+ITEMTEXT+COMMSEQ+MENUTOGGLE+
      ITEMENABLED+HIGHCOMP+CHECKED,
    0,
    (APTR)&IText4,
    NULL,
    'I',
    NULL,
    0xFFFF
};

struct IntuiText IText5 = {
    3,1,COMPLEMENT,
    20,1,
    NULL,
    "Copy Program to Disk When Changed ",
    NULL
};

struct MenuItem MenuItem1 = {
    &MenuItem2,
    0,0,
    332,10,
    CHECKIT+ITEMTEXT+COMMSEQ+MENUTOGGLE+
      ITEMENABLED+HIGHCOMP+CHECKED,
    0,
    (APTR)&IText5,
    NULL,
    'C',
    NULL,
    0xFFFF
};

struct Menu Menu4 = {
    NULL,
    261,0,
    75,0,
    MENUENABLED,
    "Options",
    &MenuItem1
};

struct IntuiText IText6 = {
    3,1,COMPLEMENT,
    1,1,
    NULL,
    "Save Defaults",
    NULL
};

struct MenuItem MenuItem11 = {
    NULL,
    0,66,
    209,10,
    ITEMTEXT+ITEMENABLED+HIGHCOMP,
    0,
    (APTR)&IText6,
    NULL,
    NULL,
    NULL,
    0xFFFF
};

struct IntuiText IText7 = {
    3,1,COMPLEMENT,
    1,1,
    NULL,
    " ",
    NULL
};

struct MenuItem MenuItem10 = {
    &MenuItem11,
    0,55,
    209,10,
    ITEMTEXT+HIGHCOMP,
    0,
    (APTR)&IText7,
    NULL,
    NULL,
    NULL,
    0xFFFF
};

struct IntuiText IText8 = {
    3,1,COMPLEMENT,
    1,1,
    NULL,
    "Name of Directory Utility ",
    NULL
};

struct MenuItem MenuItem9 = {
    &MenuItem10,
    0,44,
    209,10,
    ITEMTEXT+ITEMENABLED+HIGHCOMP,
    0,
    (APTR)&IText8,
    NULL,
    NULL,
    NULL,
    0xFFFF
};

struct IntuiText IText9 = {
    3,1,COMPLEMENT,
    1,1,
    NULL,
    "File Location - .obj",
    NULL
};

struct MenuItem MenuItem8 = {
    &MenuItem9,
    0,33,
    209,10,
    ITEMTEXT+ITEMENABLED+HIGHCOMP,
    0,
    (APTR)&IText9,
    NULL,
    NULL,
    NULL,
    0xFFFF
};

struct IntuiText IText10 = {
    3,1,COMPLEMENT,
    1,1,
    NULL,
    "Link Command",
    NULL
};

struct MenuItem MenuItem7 = {
    &MenuItem8,
    0,22,
    209,10,
    ITEMTEXT+ITEMENABLED+HIGHCOMP,
    0,
    (APTR)&IText10,
    NULL,
    NULL,
    NULL,
    0xFFFF
};

struct IntuiText IText11 = {
    3,1,COMPLEMENT,
    1,1,
    NULL,
    "Compile Command",
    NULL
};

struct MenuItem MenuItem6 = {
    &MenuItem7,
    0,11,
    209,10,
    ITEMTEXT+ITEMENABLED+HIGHCOMP,
    0,
    (APTR)&IText11,
    NULL,
    NULL,
    NULL,
    0xFFFF
};

struct IntuiText IText12 = {
    3,1,COMPLEMENT,
    1,1,
    NULL,
    "Editor Command",
    NULL
};

struct MenuItem MenuItem5 = {
    &MenuItem6,
```

(continued)

```
    0,0,
    209,10,
    ITEMTEXT+ITEMENABLED+HIGHCOMP,
    0,
    (APTR)&IText12,
    NULL,
    NULL,
    NULL,
    0xFFFF
};

struct Menu Menu3 = {
    &Menu4,
    144,0,
    111,0,
    MENUENABLED,
    "Environment",
    &MenuItem5
};

struct IntuiText IText13 = {
    3,1,COMPLEMENT,
    1,1,
    NULL,
    "Program Name to Make",
    NULL
};

struct MenuItem MenuItem18 = {
    NULL,
    0,66,
    161,10,
    ITEMTEXT+ITEMENABLED+HIGHCOMP,
    0,
    (APTR)&IText13,
    NULL,
    NULL,
    NULL,
    0xFFFF
};

struct IntuiText IText14 = {
    3,1,COMPLEMENT,
    1,1,
    NULL,
    "Obj Files to Link",
    NULL
};

struct MenuItem MenuItem17 = {
    &MenuItem18,
    0,55,
    161,10,
    ITEMTEXT+ITEMENABLED+HIGHCOMP,
    0,
    (APTR)&IText14,
    NULL,
    NULL,
    NULL,
    0xFFFF
};

struct IntuiText IText15 = {
    3,1,COMPLEMENT,
    1,1,
    NULL,
    "Run Time Flags",
    NULL
};

struct MenuItem MenuItem16 = {
    &MenuItem17,
    0,44,
    161,10,
    ITEMTEXT+ITEMENABLED+HIGHCOMP,
    0,
    (APTR)&IText15,
    NULL,
    NULL,
    NULL,
    0xFFFF
};

struct IntuiText IText16 = {
    3,1,COMPLEMENT,
    1,1,
    NULL,
    "Library Flags",
    NULL
};

struct MenuItem MenuItem15 = {
    &MenuItem16,
    0,33,
    161,10,
    ITEMTEXT+ITEMENABLED+HIGHCOMP,
    0,
    (APTR)&IText16,
    NULL,
    NULL,
    NULL,
    0xFFFF
};

struct IntuiText IText17 = {
    3,1,COMPLEMENT,
    1,1,
    NULL,
    "Linker Flags",
    NULL
};

struct MenuItem MenuItem14 = {
    &MenuItem15,
    0,22,
    161,10,
    ITEMTEXT+ITEMENABLED+HIGHCOMP,
    0,
    (APTR)&IText17,
    NULL,
    NULL,
    NULL,
    0xFFFF
};

struct IntuiText IText18 = {
    3,1,COMPLEMENT,
    1,1,
    NULL,
    "Compiler Flags",
    NULL
};

struct MenuItem MenuItem13 = {
    &MenuItem14,
    0,11,
    161,10,
    ITEMTEXT+ITEMENABLED+HIGHCOMP,
    0,
    (APTR)&IText18,
    NULL,
    NULL,
    NULL,
    0xFFFF
};

struct IntuiText IText19 = {
    3,1,COMPLEMENT,
    1,1,
    NULL,
    "Editor Flags",
    NULL
};

struct MenuItem MenuItem12 = {
    &MenuItem13,
    0,0,
    161,10,
    ITEMTEXT+ITEMENABLED+HIGHCOMP,
    0,
    (APTR)&IText19,
    NULL,
    NULL,
    NULL,
    0xFFFF
};

struct Menu Menu2 = {
    &Menu3,
    81,0,
    57,0,
    MENUENABLED,
    "Flags",
    &MenuItem12
};

struct IntuiText IText20 = {
    3,1,COMPLEMENT,
    1,1,
    NULL,
    "Quit",
    NULL
};

struct MenuItem MenuItem22 = {
    NULL,
    0,33,
    121,10,
    ITEMTEXT+COMMSEQ+ITEMENABLED+HIGHCOMP,
    0,
    (APTR)&IText20,
    NULL,
    'Q',
    NULL,
    0xFFFF
};

struct IntuiText IText21 = {
    3,1,COMPLEMENT,
    1,1,
    NULL,
    "      ",
    NULL
};

struct MenuItem MenuItem21 = {
    &MenuItem22,
    0,22,
    121,10,
    ITEMTEXT+HIGHCOMP,
    0,
    (APTR)&IText21,
    NULL,
    NULL,
    NULL,
    0xFFFF
};

struct IntuiText IText22 = {
    3,1,COMPLEMENT,
    1,1,
    NULL,
    "Use Old",
    NULL
};

struct MenuItem MenuItem20 = {
    &MenuItem21,
    0,11,
    121,10,
    ITEMTEXT+COMMSEQ+ITEMENABLED+HIGHCOMP,
    0,
    (APTR)&IText22,
    NULL,
    'O',
    NULL,
    0xFFFF
};

struct IntuiText IText23 = {
    3,1,COMPLEMENT,
    1,1,
    NULL,
    "Create New",
    NULL
};

struct MenuItem MenuItem19 = {
    &MenuItem20,
    0,0,
    121,10,
    ITEMTEXT+COMMSEQ+ITEMENABLED+HIGHCOMP,
    0,
    (APTR)&IText23,
    NULL,
    'N',
    NULL,
    0xFFFF
};

struct Menu Menu1 = {
    &Menu2,
    0,0,
    75,0,
    MENUENABLED,
    "Program",
    &MenuItem19
};

#define MenuList Menu1

struct IntuiText IText33 = {
    3,0,JAM2,
    233,156,
    NULL,
    "Run Directory Utility",
    NULL
};

struct IntuiText IText32 = {
    3,0,JAM2,
    81,186,
    NULL,
    " ",
    &IText33
};

struct IntuiText IText31 = {
    2,1,JAM2,
    11,186,
    NULL,
    "Status: ",
    &IText32
};

struct IntuiText IText30 = {
    3,0,JAM2,
    232,141,
    NULL,
    "Run Application Program",
    &IText31
};

struct IntuiText IText29 = {
    1,0,JAM2,
    78,102,
    NULL,
    "MAKE Module",
    &IText30
};

struct IntuiText IText28 = {
    1,0,JAM2,
    413,82,
    NULL,
    "Link Object Module",
    &IText29
};

struct IntuiText IText27 = {
    1,0,JAM2,
    411,64,
    NULL,
    "Compile Program",
    &IText28
};

struct IntuiText IText26 = {
    1,0,JAM2,
    77,84,
    NULL,
    "Edit Program",
    &IText27
};

struct IntuiText IText25 = {
    1,0,JAM2,
    77,64,
    NULL,
```

(continued on page 75)

90 Days to the LARGEST *AMIGA* Publication ever!

AC's GUIDE To The Commodore AMIGA®

AC's Guide to the Commodore Amiga is the only complete source for Amiga product information. AC's Guide is your only complete record of categorized and cross referenced commercial software, hardware, and freely redistributable software. AC's Guide is an Amiga reference which should be alongside every Amiga user's favorite computer. Each product is listed with a brief yet thorough description of the product's features and uses.

AC's Guide to the Commodore Amiga is available by only reservation or through your Amazing Dealer! This guide is not a part of the regular Amiga subscription. Don't risk missing this valuable AMIGA reference—reserve your AC Guide today!

Reserve your copy & save $2.00 off the $6.95 cover price!

Yes! Please enter my AC Guide reservation today!

☐ *Fall '89 $4.95 (save $2.00)* ☐ *Fall '89 & Winter '90 $8.95 (save $4.95)*

☐ *12 issues of AC plus Fall '89 & Winter '90 Guides for $32.00 (save $5.95)*

US subscriptions only. Foreign prices on request!

MasterCard VISA.

Name_____

Street_____

City_____ ST_____ Zip_____ Amount Enclosed_____

☐ Visa ☐ MC #_____ Signature _____

$20.00 minimum on all credit card orders.

All funds must be in U.S. currency drawn on a US bank. Please allow four to six weeks for processing.

```
    "Edit, Compile, Link and Execute",
    &IText26
};

struct IntuiText IText24 = {
    1,2,JAM2,
    184,31,
    NULL,
    " C Development System Options ",
    &IText25
};

#define ITextList IText24

struct NewWindow NewWindowStructure = {
    0,1,
    640,198,
    0,1,
    GADGETDOWN+MENUPICK+CLOSEWINDOW,
    WINDOWSIZING+WINDOWDRAG+WINDOWDEPTH+WINDOWCLOSE+SUPER_BITMAP+ACTIVATE,
    &full_process_gadget,
    NULL,
    " Compile - A Development Environment - c1989 CWR Systems",
    NULL,
    NULL,
    5,5,
    640,200,
    WBENCHSCREEN
};
```

Listing Two
COMPILE.c

```
#include "compile.h"
/
*****************************************************************************

                              main()

*****************************************************************************/
main()
{
int ii,jj;
char cc1,cc2;
header_ptr=0;

edit_load=0;

OpenAll();

Window=(struct Window *) OpenWindow(&NewWindowStructure);
rastport = Window->RPort;

PrintIText  (rastport,&ITextList,0,0);
SetMenuStrip(Window,&MenuList);

ii=load_env("compile");

if (ii==0)
  {
  default_environment();
  }

dir_setup();


strptr2=strptr1=getenv("INCLUDE");
strptr2--;
strcpy(include_paths[0],"");
i=1;

while  (strptr2!=NULL)
  {

  strcpy  (include_paths[i],strptr2+1);
   strptr3=find(include_paths[i],'!',1);
  if  (strptr3!=NULL)
      strptr3=strptr3-include_paths[i];

  if  (strptr3!=NULL)
      include_paths[i][strptr3]=0;

  strcat  (include_paths[i],"/");
    strcpy  (temp,include_paths[i]);
    shift_case  (temp);
    strcpy  (include_paths[i],temp);

  i++;
  strptr2=find(strptr1,'!',i);
  }


include_path_count=i;


program_name[0]=0;

print_message  ("Startup OK");

while  (1)
   {
   Wait  (1<<Window->UserPort->mp_SigBit);

   while  ( (message=(struct IntuiMessage *)
         GetMsg(Window->UserPort))  !=NULL)
      {
      message_class=message->Class;
      message_address=message->IAddress;
      message_code=message->Code;
      ReplyMsg(message);
      }

   if  (message_class==MENUPICK)
      {
      process_menus();
      }

   if  (message_class==GADGETDOWN)
      {
      process_gadgets();
      }

   if  (message_class==CLOSEWINDOW)
      {
      CloseAll();
      Exit(0);
      }

   }

}

/
*****************************************************************************

                              OpenAll()

*****************************************************************************/

OpenAll()

/* Opens the necessary Libraries */

{

IntuitionBase = (struct IntuitionBase *)
OpenLibrary("intuition.library",INTUITION_REV);

if  (IntuitionBase==NULL)
exit(FALSE);

GfxBase = (struct GfxBase *) OpenLibrary
("graphics.library",GRAPHICS_REV);

if  (GfxBase==NULL)
  {
  CloseLibrary(IntuitionBase);
  exit(FALSE);
  }
}

/
*****************************************************************************

                              CloseAll()

*****************************************************************************/

CloseAll()
{
char dirname[50];

copy_src_from_ram();
copy_exe_from_ram();
dir_get(0);
dir_cleanup();
dir_close();

CloseWindow(Window);
CloseLibrary(IntuitionBase);
CloseLibrary(GfxBase);


}

/
*****************************************************************************

                              init_pointers()

*****************************************************************************/

init_pointers()
{
int i;
for  (i=0;i<10;i++)
   argp[i]=args[i];
}


/
*****************************************************************************

                              process_menus()

*****************************************************************************/

process_menus()
{
  int kk,flgs;
   USHORT menu_number,item_number;
   char holddir[50];
   menu_number=(USHORT)MENUNUM(message_code);
   item_number=(USHORT)ITEMNUM(message_code);
   switch  (menu_number)
     {
     case 0:
      if  (program_name[0]!=0)
         {
         save_env(program_name);
         }

           switch  (item_number)
```

```
            {
            case 0:
          copy_src_from_ram();
          copy_exe_from_ram();
                get_file_name ("Input name of new Program");
          rtncd=load_env("compile");
             /* load the default environment */

          if (rtncd==0)
            {
            default_environment();
            }

          break;
             case 1:
          copy_src_from_ram();
          copy_exe_from_ram();
                get_file_name ("Select File Name to Load");
          strcpy (holddir,def_dir);
             /* Hold the directory selected by the user */
          rtncd=load_env(program_name);
             /* load the environment for this pgm */
          if (rtncd==0) rtncd=load_env("compile");
             /* if not found, use defaults */
          if (rtncd==0) default_environment();
          strcpy (def_dir,holddir);
          print_message ("starting copy to ram");
          copy_to_ram(program_name);
          flgs=PARSEINCLUDES.Flags & CHECKED;
          if (flgs!=0)
            check_for_headers();
          print_message ("OK..Program and headers loaded to VD0:");
                break;
             case 2:
                break;
             case 3:
                CloseAll();
                exit(0);
                break;
             }
    break;

case 1:
    switch (item_number)
       {
       case 0:

          input_option_data (edit_flags,"Edit Flags");
          break;

       case 1:

          input_option_data (compile_flags,"Compile Flags");
          break;

      case 2:

          input_option_data (link_flags,"Link Flags");
          break;

      case 3:

          input_option_data (link_libs,"Link Libraries");
          break;

       case 4:

          input_option_data (runtime_flags,"Runtime Flags");
          break;

       case 5:

          input_option_data (obj_to_link,"Other object files to link");
          break;

       case 6:

          input_option_data (file_to_make,"Name of File to MAKE");
          break;

    }
  break;

case 2:
    switch (item_number)
       {
       case 0:

          input_option_data (edit_program_name,"Edit Program Name");
          break;

       case 1:

          input_option_data (compile_command,"Compile Command");
          break;

      case 2:

          input_option_data (link_command,"Link Command");
          break;

      case 3:

          input_option_data (object_location,"Object File Location");
          break;

       case 4:
          input_option_data
             (directory_utility_name,"Directory Utility Name");
          break;

      case 5:
          break;
      case 6:
```



```
          print_message("Saving Defaults");
          save_env("compile");
          print_message("Ok");
          break;

          }
    break;

       }
}

/
*************************************************************************

                          input_option_data()

*************************************************************************/

input_option_data  (var_ptr,message)

char  *var_ptr,*message;
{
    char title[40];
    strcpy (title,"Input ");
    strcat (title,message);
    strcpy (RequesterWindowStructure.Title,title);
    rtncd=get_string (var_ptr);
    if (rtncd==1)
      {
      strcpy (temp,message);
      strcat (temp," Set as: ");
      strcat (temp,var_ptr);
      print_message(temp);
      }
}
/
*************************************************************************

                          process_gadgets()

*************************************************************************/


process_gadgets()

{

exit_flag=0;

if (message_address==(APTR)  &edit_gadget)
   edit();

if (message_address==(APTR)  &object_gadget)
   {
   compile();
```

(continued)

```
    if (rtncd==0)
       {
       edit_load=0;
       }
    else
       {
       edit_load=1;
       DisplayBeep(0);
       print_message
           ("There were compile errors.  Edit Program to review.");
       }
    }

if (message_address==(APTR) &make_gadget)
    {
    make();
    if (rtncd==0)
       {
       edit_load=0;
       }
    else
       {
       edit_load=1;
       DisplayBeep(0);
       print_message
           ("There were make errors.  Edit Program to review.");
       }
    }

if (message_address==(APTR) &link_gadget)
    {
    link();
    if (rtncd==0)
       {
       edit_load=0;
       }
    else
       {
       edit_load=1;
       DisplayBeep(0);
       print_message
           ("There were link errors.  Edit Program to review.");
       }
    }

if (message_address==(APTR) &run_gadget)
    run();

if (message_address==(APTR) &debug_gadget)
    debug();

if (message_address==(APTR) &directory_utility_gadget)
    directory_utility();

if (message_address==(APTR) &full_process_gadget)
    full_process();

}

/
*****************************************************************************

                            directory_utility()

*****************************************************************************/

directory_utility()
{
struct Lock *cdir,*old_dir;

cdir=Lock("sys:",ACCESS_READ);
     /* Changes to C: directory to execute the */
     /* directory utility to accommodate some  */
     /* of the quirky PD programs that must be */
     /* in a pre-defined location              */
if (cdir!=0)
    {
    old_dir=CurrentDir(cdir);
     /* retain the current directory so we can get back */

    init_pointers();
    strcpy  (args[0],directory_utility_name);
    argp[1]=0;

    new_task (" CLI Directory Utility ",1);

    CurrentDir(old_dir);  /* return ot the original directory */
    UnLock(cdir);
    }

}

/
*****************************************************************************

                            full_process()

*****************************************************************************/

full_process()
{
struct FileHandle *opfile;


opfile=Open("vd0:error.file",MODE_NEWFILE);
     /* Blow Away any error file */
Close (opfile);

exit_flag=0;
edit_load=0;

while (exit_flag==0)
   /* Any non-zero return code from compile or link */
{
   /* will keep you in this loop                    */
 if (edit_load==1)
    DisplayBeep(0);
    /* If any errors existing, beep screen */

 edit();       /* Edit the program (conditionally load error file) */
 if (rtncd!=0)
   {
   exit_flag=1;        /* If Aborted editor, get out of loop */
   }
 else
   {
   edit_load=1;
   compile();          /* Lets have a go at a compile */

   if (rtncd==0)
       {
       link();         /* If compiled OK, lets try a link */
       if (rtncd==0)
           {
           exit_flag=1;
           edit_load=0;
           run();      /* If all is well, run the sucker */
           }
       }
   }
}


/
*****************************************************************************

                            edit()

*****************************************************************************/


edit()

{

int flgs,flgs2;
init_pointers();

flgs=MULTIBUFFER.Flags & CHECKED;

strcpy  (args[0],edit_program_name);
strcpy (args[1]," ");

if (flgs!=0 || edit_load==0)
    /* If a multi buffer editor or there is */
  {
    /* no existing error file                */
  strcat  (args[1],object_location);
  strcat  (args[1],program_name);
  }

if (edit_load==1)
  strcat  (args[1]," vd0:Error.File");

argp[2]=0;

rtncd=new_task (" Editor ",0);

if (flgs==0)
  edit_load=0;  /* If not a multi buffer editor, reset flag so next */
                /* Edit will edit program and not the error file    */

if (rtncd==0)
  {
  flgs2=COPYDISK.Flags  & CHECKED;

  if ((flgs2) != 0)
    {
    copy_src_from_ram();
    }
  }


}

/
*****************************************************************************

                            compile()

*****************************************************************************/

compile()
{
char exec_name[50];

init_pointers();

strcpy  (args[0],compile_command);

strcpy  (temp,object_location);
strcat  (temp,program_name);
strcat  (temp," -O ");
strcat  (temp,object_location);
strcpy  (exec_name,program_name);
exec_name[strlen(exec_name)-2]=0;
strcat  (temp,exec_name);
strcat  (temp,".o ");
strcat  (temp,compile_flags);
strcpy  (args[1],temp);
print_message  (temp);

argp[2]=0;
```

```
rtncd=new_task(" Compile ",1);



/
*****************************************************************************

                              make()

*****************************************************************************/

make()
{
char exec_name[50];
int flgs2;

init_pointers();

strcpy  (args[0],"make");

strcpy  (exec_name,program_name);
exec_name[strlen(exec_name)-2]=0;
strcpy  (args[1],exec_name);

if  (file_to_make[0]!=0)
     strcpy (args[1],file_to_make);

argp[2]=0;

sprintf  (msg,"Now MAKING %s",args[1]);
print_message(msg);

rtncd=new_task(" Make ",1);

if  (rtncd==0)
   {
   flgs2=COPYDISK.Flags  & CHECKED;

   if  ((flgs2) != 0)
     {
     copy_exe_from_ram();
     }
   }

print_message(" ");


}

/
*****************************************************************************

                              link()

*****************************************************************************/


link()
{

char exec_name[50];
int flgs2;

init_pointers();
strcpy  (exec_name,program_name);
exec_name[strlen(exec_name)-2]=0;

strcpy  (args[0],link_command);
strcpy  (temp,link_flags);
strcat  (temp," ");
strcat  (temp,object_location);
strcat  (temp,exec_name);
strcat  (temp," ");
strcat  (temp,obj_to_link);
strcat  (temp," -O ");
strcat  (temp,object_location);
strcat  (temp,exec_name);
strcat  (temp," ");
strcat  (temp,link_libs);
strcpy  (args[1],temp);
print_message  (temp);

argp[2]=0;

rtncd=new_task(" Link ",1);

if  (rtncd==0)
   {
   flgs2=COPYDISK.Flags  & CHECKED;

   if  ((flgs2) != 0)
     {
     copy_exe_from_ram();
     }
   }

}


/
*****************************************************************************

                              run()

*****************************************************************************/


run()
{
```



```
char exec_name[50];

init_pointers();

strcpy  (exec_name,program_name);
exec_name[strlen(exec_name)-2]=0;

strcpy  (temp,object_location);
if  (file_to_make[0]!=0)
   {
   strcat  (temp,file_to_make);
   }
else
   {
   strcat  (temp,exec_name);
   }
strcpy  (args[0],temp);
strcpy  (args[1],runtime_flags);

argp[2]=0;

new_task  (" Program ",1);
}


/
*****************************************************************************

                              debug()

*****************************************************************************/


debug()
{

char exec_name[50];

init_pointers();

strcpy  (exec_name,program_name);
exec_name[strlen(exec_name)-2]=0;

strcpy  (args[0],"sdb");
strcpy  (temp,object_location);
if  (file_to_make[0]!=0)
   {
   strcat  (temp,file_to_make);
   }
else
   {
   strcat  (temp,exec_name);
   }
strcat  (temp," ");
```

(continued)

```
strcat  (temp,runtime_flags);
strcpy  (args[1],temp);

argp[2]=0;

new_task (" SDB ",1);
}


/
*************************************************************************

                        copy_to_ram()

*************************************************************************/


copy_to_ram(filename)
char *filename;
{

char
print_name[50],ram_name[50],file_to_copy[50],progline[251],header_file_name[50],*rtnstr,*first,cc;
FILE  *diskfile,*ramfile;
struct IntuiText *ipt;
int  ctr,rtn,found,ctr2,kk,flgs;
char tt[80];

strcpy  (ram_name,object_location);
strcpy  (file_to_copy,filename);
ctr=0;
first=find  (file_to_copy,'/',1);
rtnstr=first;
while (rtnstr!=0)
  {
  rtnstr=find (file_to_copy,'/',++ctr);
  }
if (ctr!=1)
  rtnstr=find (file_to_copy,'/',-ctr);  /* Find the last slash */

if (first!=NULL)
  {
  strncpy (msg,(char *) (first-7),7);
  rtn=strncmp(msg,"INCLUDE",7);
  if (rtn==0)
     rtnstr=first;
    /* Only strip the include out of path for system includes*/
  }
if (rtnstr==NULL)
  rtnstr=find(file_to_copy,':',1);

   /* if no subdir names to strip, then check for disk name to strip*/

if (rtnstr==NULL)
  rtnstr=file_to_copy-1;

strcat  (ram_name,(char *)(rtnstr+1));
strcpy  (print_name,(char *)(rtnstr+1));

diskfile=fopen(file_to_copy,"r");

ramfile=fopen(ram_name,"r");
    /* Check to see if file is already in ram */
if (diskfile!=NULL && ramfile==NULL)
  {
   fclose(diskfile);
   diskfile=NULL;


   sprintf (msg,"Copying %s to vd0:",file_to_copy);
   print_message(msg);


   diskfile=fopen(file_to_copy,"r");
   ramfile=fopen(ram_name,"w");

  while (!feof(diskfile))
       {
       get_record(progline,diskfile);
       fprintf (ramfile,"%s\n",progline);
       }
    fclose(diskfile);
    fclose(ramfile);
    print_message(" ");

   ramfile=fopen(ram_name,"r");
  }
flgs=PARSEINCLUDES.Flags & CHECKED;
if (flgs!=0)
  {
  sprintf (msg,"Now Checking %s for #include ",print_name);
  print_message(msg);
  }
while (!feof(ramfile) && flgs!=0)
   {
   get_record(progline,ramfile);
   strcat (progline,"\n");
   ctr++;

   if (progline[0]!=NULL)
      {
   /* find include directives */
      if (progline[0]=='#' && progline[1]=='i' && progline[2]=='n')
         {
         rtnstr=find(progline,'"',1);
    /* Check for beginning of the include file name */

         if (rtnstr==NULL)
            rtnstr=find(progline,'<',1);

         if (rtnstr!=NULL)
            {
            rtnstr++;
            strcpy (msg,(char *)rtnstr);
    /* Copy the include file name and terminal character to buffer */
            cc=*(rtnstr-1);
            if (cc=='<')
              cc='>';
            rtnstr=find (msg,cc,1); /* Find terminal Character */
            if (rtnstr!=NULL)
               {
               *rtnstr=NULL;   /* Eliminate terminal character */
               strcpy (header_file_name,msg);  /* Copy name */
               shift_case(header_file_name);
               found=0;
               for (ctr2=0;ctr2<header_ptr;ctr2++)
                 {
                  rtn=strcmp(header_file_name,header_files[ctr2]);
                 if (rtn==0)
                    {
                    found=1;
                    }
                 }
               if (!found)
                 {
                  strcpy (header_files[header_ptr],header_file_name);
                 header_ptr++;
                 }
               }
            }
         }
      }
   }
if (diskfile!=NULL)
  fclose (diskfile);

if (ramfile!=NULL)
  fclose (ramfile);

print_message(" ");

}


/
*************************************************************************

                        find()

*************************************************************************/

find(search_str,tgt_char,occur)

char *search_str,tgt_char;
int occur;
{
int ct,offset;
offset=ct=0;

while (*(search_str+offset)!=NULL && ct<occur)
  {
  if (*(search_str+offset)==tgt_char)
     ct++;
  offset++;
  }
offset-;

if  (*(search_str+offset)==tgt_char)
  return (search_str+offset);
else
  return (NULL);

}

/
*************************************************************************

                        check_for_headers()

*************************************************************************/


check_for_headers()

{
FILE *includefile;
char filename[50];
int ii,jj,kk,flgs;
struct IntuiText *ipt;

ii=0;

/* If include path environment variable is set with RAD:include as first
   in the string, this routine will find the file there if it exists and
   will not have to access disk as it is already in RAD: */

while  (ii<header_ptr)
  {
  for  (kk=0;kk<include_path_count;kk++)
    {
    strcpy (filename,include_paths[kk]);
    strcat (filename,header_files[ii]);
    includefile=fopen(filename,"r");
    if (includefile!=NULL)
       {
       fclose (includefile);
       copy_to_ram(filename);
       kk=99999;
       }
    }

    if (kk<99999)
     {
      sprintf (msg,"Unable to find %s",filename);
      for (kk=0;kk<include_path_count;kk++)
      DisplayBeep (NULL);
      print_message(msg);
```

```
        Delay (200);
     }

  ii++;

  }

dir_get(1);

ii=0;
while (ii<=fileptr)
   {
   jj=0;
   while (jj<header_ptr)
      {
      strcpy (filename,"RAD:INCLUDE/");
      strcat (filename,header_files[jj]);

      if (strcmp(filename,dir_files[ii])==0)
         {
         strcpy(dir_files[ii],"");
         }
      jj++;
      }
   ii++;
   }

flgs=DELETEINCLUDES.Flags & CHECKED;
if (flgs!=0)
      dir_cleanup();


}

/
*****************************************************************************

                              get_record()

*****************************************************************************/

get_record (rec,fileptr)
char *rec;
FILE *fileptr;
{
 unsigned char cc,ccc[20];
 int offset;
       offset=0;
       cc=agetc(fileptr);

       while (cc!=0xFF && cc!='\n')
         {
         *(rec+offset)=cc;
         offset++;
         cc=agetc(fileptr);
         }

            *(rec+offset)=0;

       if (cc==EOF)
          *rec='\0';
}

/
*****************************************************************************

                              copy_src_from_ram()

*****************************************************************************/


copy_src_from_ram()
{

FILE *checkfile;
char  exec_name[50],ram_exec_name[50],obj_name[50],ram_obj_name[50];
int flgs;


if (program_name[0]!=0)
   {
    save_env(program_name);
    init_pointers();
    strcpy (args[0],"copy");
    strcpy (args[1],object_location);
    strcat (args[1],program_name);
    strcpy (args[2],program_name);
    argp[3]=0;
    status_msg(program_name);
   }
}


/
*****************************************************************************

                              copy_exe_from_ram()

*****************************************************************************/


copy_exe_from_ram()
{

FILE *checkfile;
char  exec_name[50],ram_exec_name[50],obj_name[50],ram_obj_name[50];
int flgs;


if (program_name[0]!=0)
   {
    init_pointers();
    strcpy (ram_exec_name,object_location);
    strcpy (exec_name,program_name);
    exec_name[strlen(exec_name)-2]=0;
    strcat (ram_exec_name,exec_name);
    strcpy (args[0],"copy");
    argp[3]=0;

    checkfile=fopen(ram_exec_name,"r");
    if (checkfile!=NULL)
      {
      fclose (checkfile);
      strcpy (args[1],ram_exec_name);
      strcpy (args[2],exec_name);
      status_msg(ram_exec_name);
      }

    strcpy(ram_obj_name,ram_exec_name);
    strcat (ram_obj_name,".o");
    strcpy (obj_name,exec_name);
    strcat (obj_name,".o");

    checkfile=fopen(ram_obj_name,"r");
    if (checkfile!=NULL)
      {
      fclose (checkfile);
      strcpy (args[1],ram_obj_name);
      strcpy (args[2],obj_name);
      status_msg(ram_obj_name);
      }

    strcpy(ram_obj_name,ram_exec_name);
    strcat (ram_obj_name,".dbg");
    strcpy (obj_name,exec_name);
    strcat (obj_name,".dbg");

    checkfile=fopen(ram_obj_name,"r");
    if (checkfile!=NULL)
      {
      fclose (checkfile);
      strcpy (args[1],ram_obj_name);
      strcpy (args[2],obj_name);
      status_msg(ram_obj_name);
      }
   }

print_message(" ");

}


/
*****************************************************************************

                              status_msg()

*****************************************************************************/
```

(continued)

```
status_msg(name)
char *name;
{
    sprintf (msg,"Copying %s to disk",name);
    print_message (msg);
    new_task (" Copy from RAD: ",1);
}



/
*****************************************************************************
                                new_task()
*****************************************************************************/


new_task(taskname,newold)
char *taskname;
int newold;
{
int ioerror;

/* Setup argp and args and call this function */

struct FileHandle *oldcos;
struct FileHandle *opfile;
struct IntuiText *ipt;

int returncode,mode;

if (newold==1)
    mode=MODE_NEWFILE;
else
    mode=MODE_OLDFILE;

ipt=&WaitText3;
ipt->IText=taskname;


Second_Window=(struct Window *) OpenWindow(&WaitWindowStructure);
Second_rastport = Second_Window->RPort;


PrintIText  (Second_rastport,&WaitTextList,0,0);

Process=(struct Process *)FindTask (0);
oldcos=Process->pr_COS;  /* retain the output destination to restore it
later */

opfile=Open("RAD:error.file",mode);
if (opfile==0 && mode==MODE_NEWFILE)
   {
   printf ("Cannot Open File\n");
   CloseWindow (Second_Window);
   CloseAll();
   exit (99);
   }
if (opfile==0)
    opfile=Open("RAD:error.file",MODE_NEWFILE);

Process->pr_COS=opfile;
     /* Direct the output of the new task to the file */
     /* that we just opened.   This will hold all     */
     /* error messages so the editor can get 'em      */

strcpy  (temp,args[0]);
strcat  (temp," ");
strcat  (temp,args[1]);
strcat  (temp,"\n\r");

if (newold==1)
    Write  (opfile,temp,strlen(temp));


fexecv  (args[0],argp);
returncode=wait();

Process->pr_COS=oldcos;

Close  (opfile);
CloseWindow  (Second_Window);

return  (returncode);
}

/
*****************************************************************************
                                print_message()
*****************************************************************************/

print_message(msgtext)
char *msgtext;
{
struct IntuiText *ipt;
char blank_line[80];
    ipt=STATUS_LINE;
    ipt->IText=blank_line;
    strcpy  (blank_line,"
");
    blank_line[66]=0;
    PrintIText  (rastport,STATUS_LINE,0,0);
    strcpy  (blank_line,msgtext);
    blank_line[66]=0;
    PrintIText  (rastport,STATUS_LINE,0,0);
}

/
*****************************************************************************
                                get_file_name()
*****************************************************************************/

get_file_name(message)
char *message;
{
char fname[50];
int i;

for  (i=0;i<50;i++)
   strcpy  (header_files[i],"\0");

header_ptr=0;

 input_option_data  (program_name,"Program to Work On");
/*************************************************************

If you have C Heath's File requester code, comment out the
input_option_data function call above and remove the comments from
around the code below.

Link with the object for the Requester and the close_window_safely and
all will work fine

*************************************************************/

/*************************************************************
File_Window =(struct  Window  *)OpenWindow(&FileNewWindow);
Process = (struct Process *)FindTask(0L);
Main_Window = Process->pr_WindowPtr;
Process->pr_WindowPtr = Window;

get_fname(Window,NULL,message,def_program_name,def_dir);

Process->pr_WindowPtr = Main_Window;

strcpy  (fname,def_dir);
strcat  (fname,"/");
strcpy  (include_paths[0],fname);
 strcat  (fname,def_program_name);

strcpy  (program_name,fname);

strcpy  (program_name,def_program_name);

*****************************************************************************/

}

/
*****************************************************************************
                                get_string()
*****************************************************************************/

get_string(ret_string)
char *ret_string;
{

strcpy  (SIBuffer2,ret_string); /* Establishes Default String */

Second_Window=(struct  Window  *)  OpenWindow(&RequesterWindowStructure);
Second_rastport  =  Second_Window->RPort;

    if  ( IntuitionBase->lib_Version > 32 )
            {
             ActivateGadget(&string_gadget,Second_Window,0L);
            }

exit_flag=0;
while  (exit_flag==0)
   {
    Wait  (1<<Second_Window->UserPort->mp_SigBit);

    while  ( (message=(struct IntuiMessage *)
      GetMsg(Second_Window->UserPort))  !=NULL)
       {
       message_class=message->Class;
       message_address=message->IAddress;
       message_code=message->Code;
       ReplyMsg(message);
       }

    if  (message_class==GADGETDOWN)
        {

          if  (message_address==&Requester_Gadget3)
             {
             exit_flag=2;
             }

          if  (message_address==&Requester_Gadget1)
             {
              strcpy  (ret_string,SIBuffer2);
             exit_flag=1;
             }

        }

   }

    CloseWindow  (Second_Window);
    message_class=0;
    return  (exit_flag);
}


/
*****************************************************************************
                                dir_setup()
*****************************************************************************/

dir_setup()
{
```

(continued)

```
/*  Initializes Variables needed for directory acquisition */

   if ((dir_info = AllocMem((long)
    sizeof(struct FileInfoBlock),0L)) == NULL)
      {
      print_message("Unable to allocate Memory for File Info Block!");
      Delay (200);
      return(NULL);
      }

   if (FirstEntry = (struct dirent *)
    AllocMem((long)ALLOCSIZE,0L) == NULL)
      {
      print_message("Unable to allocate Memory for Directory Entry!");
      Delay (200);
      return(NULL);
      }
}


/
*****************************************************************************
                              dir_close()
*****************************************************************************/
dir_close()
{

   /* Releases the memory that was allocated to the directory buffers */


      if (FirstEntry!=NULL) FreeMem(FirstEntry,(long)ALLOCSIZE );
      if (dir_info!=NULL)   FreeMem(dir_info,
          (long)sizeof(struct  FileInfoBlock));
}


/
*****************************************************************************
                              dir_get()
*****************************************************************************/


dir_get(filter_src_obj)
int  filter_src_obj;
{
/* if filter_src_obj is set to 1 then .c and .o files are not deleted */
char incl[20];
int ctr,length;
/* Retrieves the directory contents */


print_message  ("Getting RAD: Directory");
fileptr=-1;
dirptr=0;
ctr=0;

for  (ctr=0;ctr<100;ctr++)  dir_files[ctr][0]=0;
for  (ctr=0;ctr<10;ctr++)   dir_dirs[ctr][0]=0;

strcpy  (dir_dirs[0],"RAD:INCLUDE");
ctr=0;

while(ctr<=dirptr)
  {
   if (! (pdir=(struct FileLock
*)Lock(dir_dirs[ctr],(ULONG)ACCESS_READ))  )
      {
      print_message("ERROR!!!!   Cannot find RAD:INCLUDE");
      DisplayBeep(0);
      Delay (300);
      return(99);
      }

   rtncd=Examine(pdir,dir_info);
   if (rtncd==0)
     {
     print_message("Unable to find the requested directory entry!");
     Delay(300);
     return(99);
     }

  rtncd=ExNext(pdir,dir_info);

  while (rtncd!=0)
     {
     if (dir_info->fib_DirEntryType>0)
       {
       dirptr++;
       if (dirptr>9) dirptr=9;
       strcpy (dir_dirs[dirptr],"vd0:INCLUDE/");
       strcat (dir_dirs[dirptr],dir_info->fib_FileName);
       }

     if (dir_info->fib_DirEntryType<0)
       {
       strcpy (temp,dir_info->fib_FileName);
       shift_case(temp);
       length=strlen(temp);
       if (!filter_src_obj || !(temp[length-2]=='.'
          && (temp[length-1]=='C' || temp[length-1]=='O')))

       /* Do not put C or Object files in list of files to be deleted */
         {
         fileptr++;
         if (fileptr>99) fileptr=99;
         strcpy (dir_files[fileptr],dir_dirs[ctr]);
         strcat (dir_files[fileptr],"/");
         strcat (dir_files[fileptr],dir_info->fib_FileName);
         shift_case(dir_files[fileptr]);
         }
       }

     i=ctr;
     rtncd=ExNext(pdir,dir_info);
     }
     ctr++;
  }


}


/
*****************************************************************************
                              dir_cleanup()
*****************************************************************************/

dir_cleanup()
{
int ii,jj;

for  (ii=0;ii<=fileptr;ii++)
  {
  if (dir_files[ii][0]!=0)
     {
     sprintf (msg,"Deleting %s",dir_files[ii]);
     print_message(msg);
     DeleteFile (dir_files[ii]);
     }
  }

}

/
*****************************************************************************
                              shift_case()
*****************************************************************************/

shift_case(ipstr)
char *ipstr;
{
int ii,jj;
short cc1,cc2;


jj=strlen(ipstr);

for  (ii=0;ii<jj;ii++)
  {
  cc1=*(ipstr+ii);
    if (cc1>=97 && cc1<=122)
        {
        cc2=(cc1-32);
        }
    else
        {
        cc2=cc1;
        }
  *(ipstr+ii)=cc2;
  }

}

/
*****************************************************************************
                              load_env()
*****************************************************************************/

load_env(fname)
char *fname;
{
char envname[80],flags[80];
FILE *envfileptr;
```

```
int flgs;

strcpy  (envname,fname);
strcat  (envname,".env");
envfileptr=fopen(envname,"r");

if  (envfileptr!=NULL)
   {
   print_message("Loading Environment Parameters");
   get_record  (compile_flags,envfileptr);
   get_record  (link_libs,envfileptr);
   get_record  (link_flags,envfileptr);
   get_record  (object_location,envfileptr);
   get_record  (edit_program_name,envfileptr);
   get_record  (def_program_name,envfileptr);
   get_record  (def_dir,envfileptr);
   get_record  (directory_utility_name,envfileptr);
   get_record  (compile_command,envfileptr);
   get_record  (link_command,envfileptr);
   get_record  (runtime_flags,envfileptr);
   get_record  (obj_to_link,envfileptr);
   get_record  (flags,envfileptr);
   if (strcmp(flags,"0")!=0)
      {
      COPYDISK.Flags=COPYDISK.Flags | CHECKED;
      }
   else
      {
      COPYDISK.Flags=COPYDISK.Flags & ~CHECKED;
      }
   get_record  (flags,envfileptr);
   if (strcmp(flags,"0")!=0)
      {
      PARSEINCLUDES.Flags=PARSEINCLUDES.Flags | CHECKED;
      }
   else
      {
      PARSEINCLUDES.Flags=PARSEINCLUDES.Flags & ~CHECKED;
      }
   get_record  (flags,envfileptr);
   if (strcmp(flags,"0")!=0)
      {
      MULTIBUFFER.Flags=MULTIBUFFER.Flags | CHECKED;
      }
   else
      {
      MULTIBUFFER.Flags=MULTIBUFFER.Flags & ~CHECKED;
      }
   get_record  (flags,envfileptr);
   if (strcmp(flags,"0")!=0)
      {
      DELETEINCLUDES.Flags=DELETEINCLUDES.Flags | CHECKED;
      }
   else
      {
      DELETEINCLUDES.Flags=DELETEINCLUDES.Flags & ~CHECKED;
      }
   get_record  (file_to_make,envfileptr);
   fclose  (envfileptr);
   print_message(" ");
   return  (1);
   }
else
   {
   return  (0);
   }
}

/
*************************************************************************
                              save_env()
*************************************************************************/

save_env(fname)
char  *fname;
{
char  envname[80];
FILE  *envfileptr;
int  flgs;

strcpy  (envname,fname);
strcat  (envname,".env");
envfileptr=fopen(envname,"w");

if  (envfileptr!=NULL)
   {
   print_message("Saving Environment Parameters");
   fprintf  (envfileptr,"%s\n",compile_flags);
   fprintf  (envfileptr,"%s\n",link_libs);
   fprintf  (envfileptr,"%s\n",link_flags);
   fprintf  (envfileptr,"%s\n",object_location);
   fprintf  (envfileptr,"%s\n",edit_program_name);
   fprintf  (envfileptr,"%s\n",def_program_name);
   fprintf  (envfileptr,"%s\n",def_dir);
   fprintf  (envfileptr,"%s\n",directory_utility_name);
   fprintf  (envfileptr,"%s\n",compile_command);
   fprintf  (envfileptr,"%s\n",link_command);
   fprintf  (envfileptr,"%s\n",runtime_flags);
   fprintf  (envfileptr,"%s\n",obj_to_link);
   flgs=COPYDISK.Flags & CHECKED;
   fprintf  (envfileptr,"%ld\n",flgs);
   flgs=PARSEINCLUDES.Flags & CHECKED;
   fprintf  (envfileptr,"%ld\n",flgs);
   flgs=MULTIBUFFER.Flags  & CHECKED;
   fprintf  (envfileptr,"%ld\n",flgs);
   flgs=DELETEINCLUDES.Flags  & CHECKED;
   fprintf  (envfileptr,"%ld\n",flgs);
   fprintf  (envfileptr,"%s\n",file_to_make);
   fclose  (envfileptr);
   print_message(" ");
   return  (1);
   }
else
   {
   return  (0);
   }
}

/
*************************************************************************
                              default_environment()
*************************************************************************/

default_environment()
{
   print_message("Setting Default Environment");
   strcpy  (compile_flags," -B -S +l +fi");
   strcpy  (link_libs," -lmx32 -lc32");
   strcpy  (link_flags," ");
   strcpy  (object_location,"RAD:INCLUDE/");
   strcpy  (edit_program_name,"edit");
   strcpy  (def_program_name,program_name);
   strcpy  (def_dir,"");
   strcpy  (directory_utility_name,"directutil");
   strcpy  (compile_command,"cc");
   strcpy  (link_command,"ln");
   strcpy  (runtime_flags," ");
   strcpy  (obj_to_link," ");
   strcpy  (file_to_make,"");
}
```

Listing Four
Makefile for COMPILE

```
compile:  vd0:include/compile.o  vd0:include/safeclose.o
     RAD:include/getfile.o
    ln  +Q -G RAD:include/compile.o RAD:include/safeclose.o
     RAD:include/getfile.o  RAD:lib/c32.lib
    copy RAD:include/compile.o compile.o
    copy RAD:include/compile compile
    copy RAD:include/compile.dbg compile.dbg
    del RAD:include/compile
    del RAD:include/compile.dbg

RAD:include/compile.o:  compile.c  RAD:include/compile.h
    cc compile.c -o RAD:include/compile.o -S -B +l +fi -n

RAD:include/safeclose.o:  safeclose.o
  copy safeclose.o RAD:include/

RAD:include/getfile.o:  getfile.o
  copy getfile.o RAD:include/

RAD:include/Compile.h:  compile.h
  copy compile.h RAD:include/
```

•AC•

UPS Units

C-Cor Model SW-300, Cuesta 400 Watt DataSaver & DRS-350

by Steven L. Bender

Please refer to Part I of this series (Amazing Computing, Vol 4.4) for the technical details on Switching vs. Linear Regulated Power Supplies, technical aspects and details of various types of UPS units and why they are necessary.

Amiga System Hardware

The hardware configuration I used consisted of an A-1000/512K system unit with one external A-1010 floppy disk drive, THE WEDGE hard disk interface, an MFM hard disk controller, a Thomson 4375M UltraScan multiple frequency scanning monitor, and the equivalent load. The three components were plugged into a "one to six" outlet strip plugged into the UPS unit. The external equipment was turned on together using the power switch on each UPS unit.

The Amiga A-1000 system unit /A-1010/WEDGE/DTC controller draws about around 25 Watts/35 VA; the Thomson 4375M UltraScan (adjusted with the brightness at center detent, contrast on maximum) draws 90 Watts/125 VA; and the incandescent bulb is 40 Watts/40 VA. Therefore, the load on each UPS unit tested was around 165 Watts. Since two of the three load devices use switching power supplies, the total power draw of this load is about 200 VA.

General Operating Considerations

In terms of the amplitude or voltage of the incoming AC wave, the SPS in the Amiga A-1000 and many other computers is relatively immune to minor variations in line voltage. To test exactly how immune these devices are from line voltage variations, I operated the A-1000 Amiga/Thomson 4375M combination from an AC Variac (a variable transformer) on both supranormal and subnormal AC line voltages. The most common subnormal form of line voltage is a "brownout", defined as a reduction of AC line voltage, from a nominal 117 volts to the 95 to 105 volts range.

Both the A-1000 and the Thomson UltraScan monitor functioned and operated properly (for several minutes) down to an AC line voltage of 65 volts. When no floppy drives were engaged, the A- 1000/Thomson 4375M worked properly down to an AC line voltage of about 56 volts. Below 55 volts, the UltraScan screen started to shrink, get waves, and visually distort. Eventually, as the line voltage was decreased, the A-1000 would hang. The conclusion is that these devices worked properly during brownout conditions as encountered during summertime reductions and below. However, reducing the AC line voltage to the 35 volt range for 1/10th of a second would cause the computer to hang. While most UPS units will not switch the computer over to battery backup until the AC line drops to about 105 volts, it is reassuring to know that the AC line could momentarily drop to 55 volts without adversely affecting the A-1000's normal operation .

The tests

Ten switching UPS units were tested. Each UPS was turned on without any load attached, and left to sit and charge for at least 12 hours. Then the in-use timed tests were conducted. An Amiga A-1000 system (as described above) was powered by the UPS. Kickstart and then Workbench 1.3 was loaded. PerfMon was used as the only task just to make sure the computer didn't hang. After all was set up, the UPS line cord was pulled from the wall, causing a power failure and battery backup condition. The stopwatch would be initiated, and the time would then be recorded until the CRT screen went blank and the lamp went out. Each UPS was tested in two in-use timed trials. The better of the two tests was reported in our results section.

THE UPS REVIEWS

C-Cor Model SW-300 PowerVision UPS

This standby UPS produces a true sine wave output (just like the AC line) under battery backup conditions. In contrast to most units here, this unit, compared to its power ratings, is huge. The smallest of the C-Cor UPS units is in an all-steel medium-light beige case with numerous vent holes on each side. The case is 12" wide x 6" high x 18" deep (1,296 sq in.) and weighs approximately 45 lbs. The U-shaped chassis has four welded corner beams for reinforcement. The bottom is supported by two mini "U's" bearing the bulk of the weight of the batteries and transformer. The bottom case is absolutely rigid, with no tolerance or movement possible in the bottom part. This UPS was designed in the United States and manufactured in Taiwan.

During normal operation, the green LED (AC line) on the front panel will glow, and a red LED comes on under battery backup conditions. The audible alarm is a discontinuous, rather faint beep. It can be turned off using the front panel switch. The AC line cord on the rear is permanently attached. There are no jacks to connect an external battery.

Under battery backup conditions, the audible alarm beeps every second, even after the unit has depleted its batteries and turned battery backup operations off. When the beep alarm is disabled, the battery output (red) LED still flashes. Unlike most UPS brands tested, the Power Vision's beep rate does not change significantly over time. This makes it difficult to tell whether a power outage has just started, or if the unit has been operating on battery power for half an hour, with the batteries being almost depleted.

(continued)

The unit's transformer makes a loud "THUM" when the unit is powered up. Also, there is a heavy "CLUNK" when the unit (on battery backup) switches back to the AC line. The transformer buzz is audible from over three feet away under normal AC or battery backup conditions. It appears these sounds come from the heavy windings of the transformer bucking against the steel core as the powerful AC currents interact. Industrial strength? Very close to it.

Internal, construction was nearly excellent. All wires were neatly harnessed and tied together with nylon strips. On the right side, a large, double-sided printed circuit board contained all circuitry, including 9 CMOS logic and linear IC's, an opto-isolator, and about two hundred small transistors, diodes, resistors and capacitors. There are eight, 10 turn pots on the logic board for precisely adjusting some internal voltage levels. Most of the other parts have close tolerances (e.g., 2% fixed film resistors used throughout).

There are four (MOV) varistors, three high-voltage caps, and a ferrite-cored coil in the line-filtering section. However, these parts are on the main PC board. A better design would have them absorbing energy (e.g., a lightning strike) before that energy reached the main PC Board. Most connections inside this unit use heavy-duty nylon molex connectors. One exception is the buss connector, which connects the AC high-voltage to the line filtering/protection components on the main PC board.

On the left side, a rather large 12" x 5" x 1" extruding black heat sink is attached to a pair of MJ11032 output devices. These power transistors are in modified TO-3 cases. The rated power dissipation capability (120 volt/50 amperes/300 Watts) of these two Motorola output devices totals 600 watts. Centrally located and to the rear are the four Gel-Cell batteries, each rated 6 volts/8 Amperes. There are two fuse holders mounted on the rear panel—a BLN-25 ampere battery line fuse (available from industrial lighting distributors), and the more familiar AGC-4 ampere AC line fuse.

The front mounted transformer is hefty, but is dwarfed by the heavy steel mount that holds the bank of four 8 ampere/6 volt Yuasa Gel-Cells firmly in place. I certainly consider the transformer adequate to the task, but it's too bad they couldn't find a "quiet" version, or some way of isolating it from the steel cabinet. As I mentioned, the transformer buzz was always audible when the unit was turned on. However, the temperature inside or on the case did not vary much, remaining at or near room temperature during normal AC operation, under battery backup conditions, and while charging.

The two NEMA-15 grounded outlets in the rear of this UPS unit are parallel. One wire (the neutral) is double-insulated at its plug, but at this point touches against one of the three sheet metal braces which holds the main circuit board. Because of the insulation, this is not a danger in any way, but one gathers this was an unintended problem, because the spacious interior generally has one inch "margins" between any other two component parts. There is plenty of room inside this unit; so the NEMA-15R sockets or the main circuit board could have been moved another half inch, to avoid this.

It seems the overall unit design,could have been best served if the AC line filtering / protection components had been placed on a separate circuit board near the incoming line cord and fuses, instead of right near the circuits that power the CMOS (low voltage) active circuits. This would have made for a smaller main logic circuit board, which could then have fit along the smaller dimension. This would have allowed the innards to be realigned allowing the longest dimension of the case, to be the " width ".

When there is a significant difference in the size of the width / depth dimensions, most products have the width as the longer dimension. This design topology, is shared by the Amiga A-2000, most desktop computers, and all the desks I have seen) The current C-Cor case design requires a 20+ inch depth " clearance " from the front panel, in order to plug a NEMA-15P plug into the rear AC outlet. This poses some problems for smaller desks, hutches, and computer cabinets. Most likely, this and the larger PowerVision units will be taking up floor space, but designers make decisions about size and weight. The constant timing of the audible alarm

beeps is perhaps the only fault that could have been easily changed, otherwise it is difficult to fault the PowerVision UPS.

The PowerVision comes with several " spec sheet " pages of information, but no real manual. While this information may be adequate, it is minimal.

Power Vision operates perfectly, and it is a true sinewave on battery backup. The unit is large, but conservatively designed, with almost no thermal gradients. In the first trial, the SW-300 UPS backed up the Amiga A-1000 / Thomson / equivalent load computer system for 37 minutes, 1 second. On the second in-use trial, it backed up the system for 36 minutes, 49 seconds. The sine wave output during battery backup operation wasn't as pure as that of the SAFE or Emerson units, but it was a reasonable sine wave. Overall, the unit was well designed, with one or two minor faults.

C-Cor Corporation, which originally manufactured these units in Taiwan, sold their entire stock and manufacturing rights to another company: The Power Place, just over one year ago. The Power Place upgraded the unit's power ratings , from 300 VA to 375 VA. They continue to sell all three sizes of the PowerVision units covered by their own 1 year limited warranty. The Power Place will also honor C-Cor's original 3 year limited warranty. Our sample had an early 1987 date code, but had just had new batteries installed prior to our in-use tests. List Price: $ 499.00 Unit is available discounted.

Summary

C-Cor SW-300 PowerVision Sinewave UPS

Positive attributes: Sine Wave output, conservative, efficient low-temperature design, use of an all-steel case, low price. One of the longest backup times of all UPS units tested.

Negative attributes: Large physical size and weight, lack of a significant delay upon return to the AC line, no jacks for external battery, constant timing on the audible alarm beep.

Cuesta 400 Watt (#40012060) DATASAVER UPS

The dark beige case give the impression of quality and caring in workmanship. The Cuesta 400 is one of the most compact UPS units tested here. The all-steel outer case, is 13" x 13" and only 2 1/2" high, about half the height of several of the other UPS units. This rather diminutive unit carries, which is made in the U.S.A., has a 400 watt rating, quite a powerful capability for its size.

In normal operation, both of the front panel LED indicators glow. One glows green, indicating normal AC line operation, the other one glows yellow, when there is power at the rear outlets. Under battery backup conditions, the green LED turns red, and flashes. Each time the red LED flashes, the audible alarm beeps.

There is no external switch to turn the beep alarm off, but the buzzer can be disabled by inserting a jumper internally. The unit beeps every five seconds when the batteries are fully charged. That beep rate decreases to once every second after about 10 minutes of battery backup under the load of the Amiga hardware. At some point, the unit just reaches a predetermined point of battery depletion, and the AC Power LED goes off. If the AC line is still out, the Cuesta just goes to sleep, and the computer goes off. After the AC line returns, the LED (now green) flashes for a while, telling you the battery was depleted and is now charging.

Internal construction was excellent, using a double sided PC Board for the CMOS logic circuitry. There were nine IC's (mostly CMOS) two opto-isolators, many small transistors, two TO-220 plastic power transistors, one regulator. A high quality torroidal transformer is centrally mounted in this neatly laid out package. The transformer buzz was inaudible, but the transformer wasn't screwed in tightly. The all-steel case consists of a bottom and top units, which electro-magnetically shields the torroid, but causes it to vibrate very slightly. Construction is definitely: heavy duty. Two stud type power transistors are present, and they are visible from the rear, being mounted on a large extruded black heat sink, which is thermally linked to the inner case. Those output transistors stayed fairly cool to the touch.

(continued)

In fact, the entire unit stayed quite cool, during normal AC operation, battery backup, and battery charging conditions.

There are three NEMA-15 grounded outlets in the rear this UPS unit; three for UPS power, and ne for line only (it goes dead on battery backup). There are no external fuses, and one internal 30 Ampere battery fuse . Two hidden jacks connect an external 24 Volt battery to the unit. An internal jack can be wired for an external alarm signal. The available AC power, is limited by a 5 Ampere AC line input circuit breaker on the rear panel, and the AC line cord is permanently attached. The Cuesta 400 unit has two MOV's and a high speed Transorb surge suppresser, I found found little evidence of capacitors, coils or resistors that I could attribute to its line filtering capabilities. Perhaps part of the torroid is used in this manner. All the other UPS units tested here have some obvious line filtering capabilities inside.

The 12 page manual that came with the 400 watt Datasaver was comprehensive, including graphs, charts, schematic and more technical data than most people would need or want. There was also one page Quick Reference Card.

The Cuesta 400 UPS backed up the expanded Amiga A-1000 / Thomson / equivalent expanded computer system, for 27 minutes, 6 seconds.

This unit deserves kudos for its very compact size, and well thought out thermal design. List Price: $ 695.00 Unit is available discounted.

Summary

CUESTA 400 Watt (#40012060) DATASAVER UPS

Positive attributes: Small , with low-temperature design, all steel case, external battery jacks, UR certified.

Negative attributes: High list price, lack of significant line noise filtering, less backup time capabilities than several other UPS units, no alarm turn-off switch, lack of a delay upon return to the AC Line.

DRS POWER SYSTEMS DRS-350

The DRS-350 has a 350 Watt rating, and 600 Watt, and 1KW versions are also available. The DRS-350 is designed and made in the U.S. The case is heavy gauge aluminum measuring 9" wide x 6.5" high x 16.5" deep; the unit is unusually deep. The front panel is small, but it contains a rather large assortment of LED indicators. On the left is the " LOAD CENTER " display. This is made up of two columns of LEDs indicating both battery condition, and the AC load connected to the unit.

The power switch is located on the lower right hand side of the front panel. Two separate LEDs located above the power switch indicate that the UPS is either in " normal " or " standby " condition. When AC power is present, the battery display shows a single LED lit for " battery " level, and if a load is connected, the load display also shows a single LED lit for the " load " level. Upon going to battery backup, each column of ten LEDs turns into a bar graph meter display for battery capacity, and AC load level.

The rear panel has four NEMA-15 AC outlets, two fuses (not labeled), and a CEE-22 grounded power cord receptacle. The fuse on the left is in series with the AC line; the more centrally located fuse appears to be in series with the four rear mounted AC outlets. When you turn the unit on, the multicolored battery meter display comes to life. The unit normally stays in battery backup condition for a second or so, then goes to normal condition. The DRS-350 is designed so it will not cold boot and output battery power if AC power is absent at turn-on. Under that condition, the indicators on the battery meter all light, but none of the load meter LEDs light. Also, there is no hum from the transformer to indicate either battery charging or battery / inverter operation.

An audible alarm sounds under backup conditions, but it is quite faint . The audible alarm initially beeps every 4 seconds after AC power is lost, and there is no switch to disable it. This beep becomes a continuous whine when the uppermost six or seven of the battery meter LEDs are extinguished (as the battery is depleted).

At that point the orange LED is still lit and the backup time is running out. A battery depletion circuit protects the batteries,by engaging a UPS shut-down when the batteries reach a predetermined low voltage. On both samples tested, three of the battery meter LEDs remained lit, (both red LEDs and the yellow LED were lit) when the unit shut down. The documentation states that the UPS should last until the last red battery meter LED is lit before battery depletion shutdown. This is a minor discrepancy, which might be affected by the size of the load present on the UPS.

Inside, the power transformer is not that large compared to the single battery, but it does seem adequate to the task. The power transformer has a smaller core than any other UPS unit tested, except the Meirick UPS. Like the noisy transformer in the Meirick unit, the DRS-350 exhibited a significant amount of audible hum, noise and vibration during normal operation. This noise level became even louder when the unit switched to battery backup. It was quite difficult to hear the audible alarm on the DRS unit, over the transformer hum.

The four NEMA-15 outlets are not directly connected to a surge protecting parts, but 3 MOV's are present on the main PC board. There is a single stage input RFI/EMI line filter on the AC line cord socket. The spike/surge protection should be good in this unit. Inside, the unit has sort of an empty look—they could have re-engineered it and knocked of about two inches on the depth.

The unit is well made, with circuitry on two high quality PC Boards. The unit uses 6 IC's, including 3 on the main PC board and 3 for the LED drivers. I noted no opto-isolators. The main output devices are four plastic (TIP-35C) power transistors mounted on a rather substantial aluminum heat sink thermally linked to the main body (U section) of the outer case. This heatsink covers and encases the gel cell battery. The rated power (100 volts / 25 Ampere / 200 Watt) capabilities of the four Motorola output devices totals: 800 Watts.

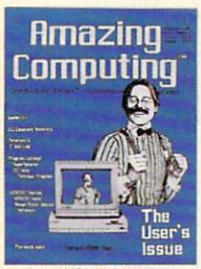
It was therefore, somewhat disappointing, that our first sample of the DRS-350 had a significant defect in its LED display. This didn't show up the first time it was turned on, but it did show up the second time. On normal AC operation, this sample also gave off a slight burning smell, and seemed to make too much internal noise. The unit was not used for our in-use tests.

I got a second DRS-350 and tested it. This unit had a different defect: it always indicated it was in " battery " operation, with the red LED lit. In addition, the displays were always in bar graph mode, even when it was running on AC power. This defect occurred from the first time it was turned on. The

(continued on page 105)

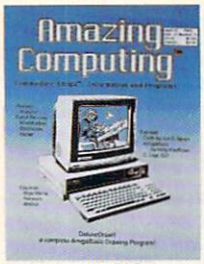
Amazing COMPUTING™

• E X P A N D I N G R E F E R E N C E •



VOLUME 1.1

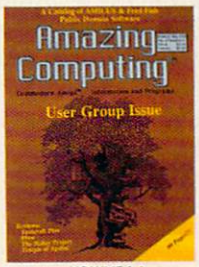
VOLUME 1.2



VOLUME 1.3

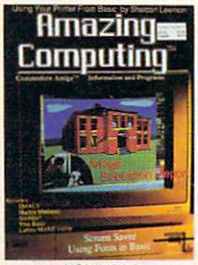
VOLUME 1.4

VOLUME 1.5



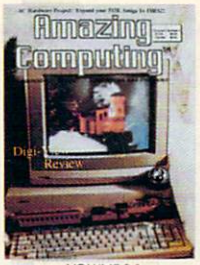
VOLUME 1.6

VOLUME 1.7



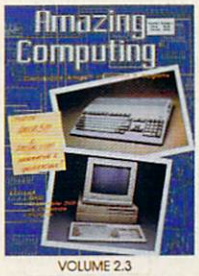
VOLUME 1.8

VOLUME 1.9

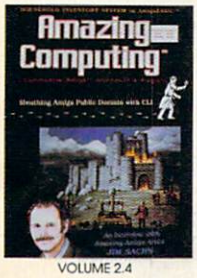


VOLUME 2.1

VOLUME 2.2



VOLUME 2.3

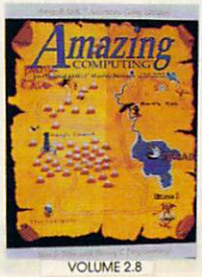


VOLUME 2.4

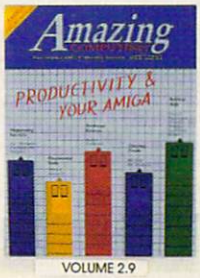
VOLUME 2.5

VOLUME 2.6

VOLUME 2.7



VOLUME 2.8

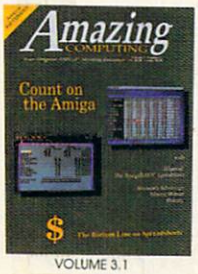


VOLUME 2.9

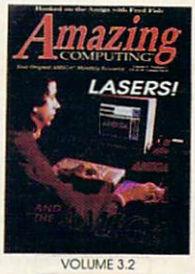
VOLUME 2.10

VOLUME 2.11

VOLUME 2.12



VOLUME 3.1



VOLUME 3.2

VOLUME 3.3

VOLUME 3.4

VOLUME 3.5

VOLUME 3.6

VOLUME 3.7

VOLUME 3.8

VOLUME 3.9

Amazing COMPUTING™

Expanding Reference

Expanding reference is not just an empty promise. The pages of Amazing Computing™ are filled with articles on technical operations and procedures, basic use, and just-plain-fun. The growing library of Amazing Computing's Back Issues contains articles ranging from building your own IBM Disk controller, to setting up your own startup sequence. Amazing Computing™ has repeatedly been the first magazine to offer the Amiga users solid, in depth reviews and hands on articles for their machines.

From the Beginning

Since February 1986, Amazing Computing™ has been providing users with complete information for their Amigas. This store house of programs and information is still available through our back issues. From the Premiere issue to the present, there are insights into the Amiga any user will find useful. AC was the first magazine to document CLI, tell its readers how to connect a 5 1/4 IBM drive, describe a 1 meg upgrade hardware project for the A1000, and many more. Please read the list of topics AC has covered below to find the information you have been missing.

Back Issues are $5.00 US, $6.00 Canada and Mexico, $7.00 Foreign Surface

All payments must be made by check or money order in U.S. funds drawn on a U.S. Bank.

Limited Supply

Unfortunately, nothing lasts forever, and the availability of some of our Back Issues is definitely limited. Complete your Amazing Computing™ library today, while these issues are still available, by completing the order form in the back of this issue.

Volume 1 Number 1 Premiere 1986

Super Spheres By Kelly Kauffman An ABasic Graphics prog.
Date Virus By J Foust A disease may attack your Amiga!
EZ-Term by Kelly Kauffman An ABasic Terminal program
Miga Mania by P. Kivolowitz Programming fixes & mouse care
Inside CLI by G. Musser Jr. A list of CLI commands
AmigaForum by B. Lubkin Visit Compuserve's Amiga SIG
Commodore Amiga Development Program by D. Hicks
Amiga Products A listing of present and expected products

Volume 1 Number 2 March 1986

Electronic Arts Comes Through A review of software from EA
Inside CLI: part two G. Musser Investigates CLI & ED
A Summary of ED Commands
Live! by Rich Miner A review of the Beta version of Live!
Online and the CTS Fabite 2424 ADH Modem by J. Foust
Superterm V 1.0 By K. Kauffman A term. prog. in Amiga Basic
A Workbench "More" Program by Rick Wirch
Amiga BBS numbers

Volume 1 Number 3 April 1986

Analyze! a review by Ernest Viverios
Reviews of Racter, Barataccas and Mindshadow
Forth! The first of our on-going tutorial
Deluxe Draw!! by R. Wirch An Amiga Basic art program
Amiga Basic, A beginners tutorial
Inside CLI: part 3 by George Musser George gives us PIPE

Volume 1 Number 4 May 1986

SkyFox and Articfox Reviewed
Build your own 5 1/4 Drive Connector By Ernest Viveiros
Amiga Basic Tips by Rich Wirch
Scrimper Part One by P. Kivolowitz prog to print Amiga screen
Microsoft CD ROM Conference by Jim O'Keane
Amiga BBS Numbers

Volume 1 Number 5 1986

The HSI to RGB Conversion Tool
by S. Pietrowicz Color manipulation in BASIC
AmigaNotes by Rick Rae The first of the Amiga music columns
Sidecar A First Look by John Foust A first "under the hood"
John Foust Talks with R. J. Mical at COMDEX™
How does Sidecar affect the Transformer
an interview with Douglas Wyman of Simile
The Commodore Layoffs by J. Foust A look Commodore "cuts"
Scrimper Part Two by Perry Kivolowitz
Marauder reviewed by Rick Wirch
Building Tools by Daniel Kary

Volume 1 Number 6 1986

Temple of Apshai Triology reviewd by Stephen Pietrowicz
The Halley Project: A Mission reviewed by S Pietrowicz
Flow: reviewed by Erv Bobo
Textcraft Plus a First Look by Joe Lowery
How to start your own Amiga User Group by William Simpson
Amiga User Groups
Mailing List by Kelly Kauffman a basic mail list program
Pointer Image Editor by Stephen Pietrowicz
Scrimper: part three by Perry Kivolowitz
Fun With the Amiga Disk Controller by Thom Sterling
Optimize Your AmigaBasic Programs for Speed by Pietrowicz

Volume 1 Number 7 1986

Aegis Draw: CAD comes to the Amiga by Kelly Adams
Try 3D by Jim Meadows an introduction to 3D graphics
Aegis Images/ Animator: a review by Erv Bobo
Deluxe Video Construction Set reviewed by Joe Lowery
Window requesters in Amiga Basic by Steve Michel
ROT by Colin French a 3D graphics editor
"I C What I Think" Ron Peterson with a few C graphic progs
Your Menu Sir! by B Catley program Amiga Basic menues
IFF Brush to AmigaBasic 'BOB' Basic editor by M Swinger
Linking C Programs with Assembler Routines...by Gerald Hull

Volume 1 Number 8 1986

The University Amiga By G.Gamble Amiga at Washington State
MicroEd a look at a one man army for the Amiga
MicroEd, The Lewis and Clark Expedition reviewed Frizelle
Scribble Version 2.0 a review
Computers in the Classroom by Robert Frizelle
Two for Study by Frizelle Discovery & TheTalking Coloring Book
True Basic reviewed by Brad Grier
Using your printer with the Amiga
Marble Madness reviewed by Stephen Pietrowicz
Using Fonts from AmigaBasic by Tim Jones
Screen SaVer by P. Kivolowitz A monitor protection prog. in C
Lattice MAKE Utility reviewed by Scott P. Evernden
A Tale of Three EMACS by Steve Poling
.bmap File Reader in Amiga Basic by T Jones

Volume 1 Number 9 1986

Instant Music Reviewed by Steve Pietrowicz
Mindwalker Reviewed by Richard Knepper
The Alegra Memory Board Reviewed by Rich Wirch
TxEd Reviewed by Jan and Cliff Kent
Amazing Directory A guide to the sources and resources
Amiga Developers A listing of Suppliers and Developers
Public Domain Catalog A listing of Amicus and Fred Fish PDS
Dos 2 Dos review R. Knepper Transfer files from PC/MS-DOS
MaxiPlan review by Richard Knepper The Amiga Spreadsheet
Gizmoz by reviewed by Peter Wayner Amiga extras!
The Loan Information Program by Brian Catley
basic prog. to for your financial options
Starting Your Own Amiga Related Business by W. Simpson
Keep Track of Your Business Usage for Taxes by J. Kummer
The Absoft Amiga Fortran Compiler reviewed by R A. Reale
Using Fonts from AmigaBasic, Part Two by Tim Jones
68000 Macros on the Amiga by G. Hull Advance your ability.
TDI Modla-2 Amiga Compiler review by S Faiwisze

Volume 2 Number 1 1987

What Digi-View Is... Or, What Genlock Should Be! by J. Foust
AmigaBasic Default Colors by Bryan Catley
AmigaBasic Titles by Bryan Catley
A Public Domain Modula-2 System reviewed by Warren Block
One Drive Compile by Douglas Lovell Lattice C with one drive
A Megabyte Without Megabucks by Chris Irving
An Internal Megabyte upgrade
Digi-View reviewed by Ed Jakober
Defender of the Crown reviewed by Keith Conforti
Leader Board reviewed by Chuck Raudonis
Roundhill Computer System's PANEL reviewed by Ray Lance
Digi-Paint.....by New Tek previewed by John Foust
Deluxe Paint II ....from Electronic Arts previewed by J. Foust

Volume 2 Number 2 1987

The Modem by Josph L. Rothman efforts of a BBS Sysop
MacroModem reviewed by Stephen R. Pietrowicz
GEMINI or "It takes two to Tango" by Jim Meadows
Gaming between machines
BBS-PC! reviewed by Stephen R. Pietrowicz
The Trouble with Xmodem by Joseph L. Rothman
The ACO Project....Graphic Teleconferencing on the Amiga
by S. R. Pietrowicz
Flight Simulator II....A Cros Country Tutorial by John Rafferty
A Disk Librarian in AmigaBASIC by John Kennan
Creating and Using Amiga Workbench Icons by C. Hansel
AmigaDOS version 1.2 by Clifford Kent
The Amazing MIDI Interface build your own by Richard Rae
AmigaDOS Operating System Calls and
Disk File Management by D. Haynie
Working with the Workbench by Louis A. Mamakos Prog in C

Volume 2 Number 3

The Amiga 2000™ by J Foust
A First look at the new, high end Amiga™
The Amiga 500™ by John Foust
A look at the new, low priced Amiga
An Analysis of the New Amiga PCs by J. Foust
Speculation on the New Amigas
Gemini Part II by Jim Meadows
The concluding article on two-player games
Subscripts and Superscripts in AmigaBASIC by Ivan C. Smith
The Winter Consumer Electronics Show by John Foust
AmigaTrix by W. Block Amiga™ shortcuts
Intuition Gadgets by Harriet Maybeck Tolly
A journey through gadget-land, using C
Shanghai reviewed by Keith M. Conforti
Chessmaster 2000 & Chessmate reviewedby Edwin V. Apel, Jr.
Zing! from Meridian Software reviewed by Ed Bercovitz
Forth! by Jon Bryan Get stereo sound into your Forth programs.
Assembly Language on the Amiga™ by Chris Martin
Roomers by theBandito Genlocks are finally shipping, & MORE!!!
AmigaNotes by R. Rae Hum Busters... "No stereo? Y not?...
The AMICUS Network by J. Foust
CES, user group issues and Amiga Expo'

Volume 2 Number 4 1987

Amazing Interviews Jim Sachs by S. Hull Amiga Artist
The Mouse That Got Restored by Jerry Hull and Bob Rhode
Sluething Public Domain Disks with CLI by John Foust
Highlights: the San Francisco Commodore Show by S Hull
Speaker Sessions: San Francisco Commodore Show H Tolly
Household Inventory System in AmigaBASIC™ by B Catley
Secrets of Screen Dumps by Natkun Okun
Using Function Keys with MicroEmacs by Greg Douglas
Amigatrix II by Warren Block More Amiga shortcuts
Basic Gadgets by Brian Catley Create gadget functions
Gridiron reviewed by K. Conforti Real football for the Amiga
Star Fleet I Version 2.1 reviewed by J. Tracy Amigain Space
The TIC reviewed by J. Foust Battery powered Clock Calendar
Metascope review by H. Tolly An easy-to-use debugger

Volume 2 Number 5 1987

The Perfect Sound Digitizer review by R. Battle
The Future Sound Digitizer by W. Block Applied Vision's SD
Forth! by J. Bryancomparing JForth amd Multi-Forth.
Basic Input by B. Catley AmigaBASIC input routine for use in
all your programs.

Volume 2 Number 5 1987 continued

Writing a SoundScape Module in C by T. Fay Programming with MIDI, Amiga and SoundScape by SoundScape author.
Programming in 68000 Assembly Language by C. Martin
Continuing with Counters & Addressing Modes.
Using FutureSound with AmigaBASIC by J. Meadows
AmigaBASIC Programming utility with real, digitized STEREO
AmigaNotes Rich Rae reviews SoundScape Sound Sampler.
More AmigaNotes by R. Rae A furtherlook at Perfect Sound .
Waveform Workshop in AmigaBASIC by J. Shields edit & save
waveform for use in other AmigaBASIC programs.
The Mimetics Pro MIDI Studio by Sullivan, Jeffery
A review of Mimetics' music editor/player.
Intuition Gadgets Part II by H. MaybeckTolly Boolean gadgets
provide the user with an on/off user interface.

Volume 2 Number 6 1987

Forth! by J. Bryan Access resources in the ROM Kernal.
The Amazing Computing Hard Disk Review by J. Foust & S. Leemon In-depth looks at the C Ltd. Hard Drive, Microbotics' MAS-Drive20, Byte by Byte's PAL Jr., Supra's 4x4 Hard Drive and Xebec's 9720H Hard Drive. Also. a look at disk driver software currently under devlopment.
Modula-2 AmigaDOS™ Utilities by S. Faiwiszewsk A
Calls to AmigaDOS and the ROM kernal.
Amiga Expansion Peripheral by J. Foust
Explanation of Amiga expansion peripherals.
Amiga Technical Support by J. Foust
How and where to get Amiga tech support.
Goodbye Los Gatos by J. Foust Closing Los Gatos.
The Amicus Network by J. Foust West Coast Computer Faire.
Metacomco Shell and Toolkit by J. Foust A review
The Magic Sac by J. Foust Run Mac programs on your Amiga.
What You Should Know Before Choosing an Amiga 1000
Expansion Device by S. Grant
7 Assemblers for the Amiga by G. Hull Choose your assembler
Shakeup Replaces Top Management at Commodore by S. Hull
Peter J. Baczor"by S. Hull Manager at CBM gives an inside look
Logistix A review by Richard Knepper
Organize! by A review Richard Knepper database.
68000 Assembly Language Programming on the Amiga
by Chris Martin
Superbase Personal Relational Database by Ray McCabe
AmigaNotes by Rae, Richard A look at FutureSound
Commodore Shows the Amiga 2000 and 500 at the Boston
Computer Society by H Maybeck Tolly

Volume 2, Number 7 1987

New Breed of Video Products by John Foust...
Very Vivid! by Tim Grantham...
Video and Your Amiga by Oran Sands III
Amigas & Weather Forecasting by Brenden Larson
A-Squared and the Live! Video Digitizer by John Foust.
Aegis Animator Scripts and Cel Animation by John Foust
Quality Video from a Quality Computer by Oran Sands III.
Is IFF Really a Standard? by John Foust..
Amazing Stories and the Amiga™ by John Foust.
All about Printer Drivers by Richard Bielak
Intuition Gadgets by Harriet Maybeck Tolley.
Deluxe Video 1.2 by Bob Eller
Pro Video CG1 by Oran Sands III.
Digi-View 2.0 Digitizer/Software by Jennifer M. Janik
Prism HAM Editor from Impulse by Jennifer M. Janik

Volume 2, Number 7 1987 continued

Easy! drawing tablet by John Foust..
CSA's Turbo-Amiga Tower by Alfred Aburto
68000 Assembly Language by Chris Martin.

Volume 2, Number 8 1987

This month Amazing Computing™ focuses on entertainment packages for the Amiga. Amazing game reviews...SDI, Earl Weaver Baseball, Portal, The Surgeon, Little Computer People, Sinbad, StarGlider, King's Quest I,II and III, Faery Tale Adventure, Ultima III, Facets of Adventure, Video Vegas and Bard's Tale.
Plus Amazing monthly columns... Amiga Notes, Roomers, Modula-2, 68000 Assembly Language and The Amicus Network.
Disk-2-Disk by Matthew Leeds
The ColorFonts Standard by John Foust
Skinny C Programs by Robert Riemersma, Jr.
Hidden Messages In Your Amiga™ by John Foust
The Consumer Electronics Show and Comdex.by J Foust

Volume 2 Number 9 1987

Analyze 2.0 reviewed by Kim Schaffer
Impact Business Graphics review by Chuck Raudonis
Microfiche Filer review by Harv Laser
Pagesetter review by Rick Wirch
Gizmoz Productivity Set 2.0 review by Bob Eller
Kickwork review by Harv Laser
Diga Telecommunications Package review by Steve Hull
Mouse Time and Timesaver review by John Foust
Insider Memory Expansion review by James O'Keane
Microbotics Starboard-2 review by S. Faiwiszewski
Leather Goddesss of Phobos by Harriet Maybeck-Tolly
Lattice C Compiler Version 3.10 reviewed by Gary Sarff
Manx 3.4a Update reviewed by John Foust
AC-BASIC reviewed by Sheldon Leemon
AC-BASIC Compilier an alternative comparison by B Catley
Modula-2 Programming S Faiwiszewski Raw Console Dev. Events
Directory Listings Under AmigaDOS by Dave Haynie
AmigaBASIC Patterns by Brian Catley
Programming with Soundscape Todor Fay manipulate's samples
Bill Volk, Vice-President Aegis Development, by Steve Hull
Jim Goodnow, Devekoper of Manx 'C' interview by Harriet M Tolly
Plus a great collection of monthly columns...

Volume 2 Number 10 1987

Max Headroom and the Amiga by John Foust
Taking the Perfect Screen Shot by Keith Conforti
Amiga Artist: Brian Williams by John Foust
Amiga Forum on Compuserve™... Software Publishing Conference Transcript by Richard Rae
All About Online Conferencing by Richard Rae
dBMAN reviewed by Clifford Kent
Amiga Pascal reviewed by Michael McNeil
AC-BASIC Compiler reviewed by Bryan Catley
68000 Assembly Language by Chris Martin
Amiga Programming:
Amiga BASIC Structures by Steve Michel
Quick and Dirty Bobs by Michael Swinger
Directory Listings Under Amiga-DOS, Part II by Dave Haynie
Fast File I/O with Modula-2 by Steve Faiwiszewski
Window I/O by Read Predmore
Plus a great collection of monthly columns...

Volume 2 Number 11 1987

Word Processors Rundown by Geoff Gamble
ProWrite, Scribble!, and WordPerfect compared
LPD Writer Review by Marion Deland
VizaWrite Review by Harv Laser
Aedit Review by Warren Block
WordPerfect Preview by Harv Laser
Jez San Interview by Ed Bercovitz—StarGlider author speaks!
Do-it-yourself Improvements to the Amiga Genlock
Digi-Paint Review by Harv Laser
Sculpt 3D Review by Steve Pietrowicz
Shadowgate Review by Linda Kaplan
TeleGames Review by Michael T. Cabral
Reason Preview: an intense grammar examination application
As I See It by Eddie Churchill WordPerfect,Gizmoz V2.0 and Zing!
AmigaNotes by R Rae 4 electronic music books
Modula-2 Programming by S.Faiwiszewski devices, I/O, &serial port
68000 Assembly Language by Chris Martin Display routines
The AMICUS Network by John Foust—Desktop Publishing, Seybold
C Animation Part II by Mike Swinger Animation Objects
BASIC Text by Brian Catley Pixel perfect text positioning
Soundscape Part III by Todor Fay VU Meter and more
Fun with Amiga Numbers by Alan Barnett
File Browser by Bryan Catley—Full Feature BASIC File Browsing
Plus a great collection of monthly columns...

Volume 2 Number 12 1987

The Ultimate Video Accessory by **Larry White**
The Sony Connection by **Stewart Cobb**
15-Puzzle in AmigaBASIC by Zoltan Szepsi
Life, Part I: The Beginning by **Gerald Hull**
The ultra-complex nine bit solution to the "Game of Life."
Amiga Virus! by John Foust
CLI Arguments in C by Paul Castonguay
MIDI Interface Adapter by Barry Massoni
Amiga 1000-style MIDI interfaces can fit A2000s or 500s
Modula-2 by S. Faiwiszewski Part 1: command line calculator
AmigaNotes by Rick Rae audio changes made in the A500 &A2000.
Animation for C Rookies: Part III by M. Swinger double-buffering.
The Big Picture by Warren Ring Assembly language programming
Karate Kid Review by Stephen R. Pietriowicz
GO! 64 review by John Foust, James O'Keane, and Rick Wirch
Three C-64 experts investigate a new Amiga 64 emulator.
A-Talk-Plus Review by Brendan Larson
Calligrapher Review by John Foust
Animator: Apprentice Review by John Foust
Playing Dynamic Drums on the Amiga by David N. Blank
WordPerfect Review by Steve Hul
Insider/Kwikstart Review by Ernest P. Viveiros Sr
RAM & ROM expansion: Comments and installation tips.
Forth! by Jon Bryan DumpRPort utility for your Multi-Forth toolbox.
As I See It by Eddie Churchill Digi-Paint, Portal, &Videoscape 3D.
The Commodore Show and AmiExpo: New York!
Plus a great collection of monthly columns...

Volume 3 Number 1 1988

AmigaNotes by Richard Rae Amiga digital music generation .
C Animation Part IV by Michael Swinger
Forth by John Bryan Sorting out Amiga CHIP and FAST memory
The Big Picture by Warren Ring Daring assembler language programming; CLI system calls and manipulating disk files.

Volume 3 Number 1 1988 continued

68000 ASsembly Langueage Programming by Chris Martin "Create a multi-color screen without using Intuition routines!"
Modula-2 Programming by S. Faiwiszewski A new modula-2!
Amicus Network Special Report: Fall COMDEX by J. Foust
The ultimate Video Accessory: Part II by Larry White
Life: Part II by Gerald Hull The Amiga blitter."
FormatMaster: Professional Disk Formatting Engine byC.Mann Put Batch language to work on the drudgery of disk formatting .
BSpread by Brian Catley full featured AmigaBASIC spreadsheet!
AmigaForum Transcript ed. by Rick Rae Amiga's Dave Haynie.
Haicalc Review by Chuck Raudonis easy to use, spreadsheet.
VIP Professional Review by S. Mitchell Manage stock portfolio
Money Mentor Review by S.Kemp Personal finance system.
Investor's Advantage Review by Richard Knepper
plus "Poor Man's Guide to the Stock Market."
Plus a great collection of monthly columns...

Volume 3 Number 2 1988

Laser Light Shows with the Amiga by Patrick Murphy
Lasers and the Amiga: A Dazzling Tandem
The Ultimate Video Accessory: PartIII by Larry White
Take the final stepstoward designing your own videos.
Our First Desktop Video by Larry White
Step-by -step guide to organizing & presenting your Amiga video.
Hooked on the Amiga with Fred Fish interview by Ed Berkovitz.
Photo Quality Reproduction with the Amiga and Digi-View by Stephen Lebans
Balancing your Checkbook with WordPerfect Macros by S.Hull
Hand your checkbook worries over to the Amiga.
More Basic Text by Bryan Catley easier text on an Amiga screen
Life: Part III by Gerald Hull
Sries winds up with famed nine-bit calculation & source to LIFER.
Solutions to Linear Algebra through Matrix Computations by Ropbert Ellis Simplify matrix algebra basic operations & routines.
Modula-2 Programming by Steve Faiwiszewski
Catching up with Calc-a source follow-up.
68000 Assembler Language Programming by Chris Martin Graphics- Part II of Assemgram.
Arazok's Tomb interview by Kenneth E. Schaefer
AiRT by S. Faiwiszewski innovative icon-basede program. lang.
Forms in Flight by S. Pietrowicz Render & Animate 3D objects
Silicon Dreams and the Jewel of Darkness by K E. Schaefer
Leisure suit Larry by Kenneth E. Schaefer
Two New Entries From Microbiotics by John Foust
M501 Expansion & Starboard II MultiFunctionboard.
Mindlight 7 and People Meter by John Foust
Phantasie Ken E. Schaefer Amazing Phantasie Character Editor.
Plus a great collection of monthly columns...

Volume 3 Number 3 1988

Desktop Video,Part IV by Larry White
Put all the pieces together-the desktop video commercial.
The Hidden Power of CLI Batch File Processing by J. Rothman
Make your Amiga easier to use with CLI Batch files.
A Conference with Eric Graham edited by John Foust
The mastermind behind Sculpt 3D and Animate 3D.
Perry Kivolowitz Interviewed by Ed Bercovitz Amiga insights from a major developer and personality.
Jean "Moebius" Giraud Interviewed by Edward L. Fadigan
Avant-garde art comes to the Amiga-in dazzling form.
PAL Help by Perry Kivolowitz A1000 expansion reliability.
Boolean Function Minimization by Steven M. Hart
A useful digital design tool in AmigaBASIC.
Amiga Serial Port and Midi Compatibility for Your A2000! by L. Ritter and G. Rentz Add an A1000-style serial port to the A2000!
Electric Network Solutions the Matrix Way by Robert Ellis
Engineers! Practicle routines for using matrix algebra.
The A.M.U.G. BBS List compiled by Joe Rothman, Chet Solace, & Dorothy Dean 514 BBS phone numbers in the U.S. & Canada.
FACC II reviewed by Graham Kinsey Speed your floppy drives.
Uninvited reviewed by K E. Schaefer
Flow reviewed by Pamela Rothman brainstorms into mental art.
Benchmark Modula-2 Compiler reviewed by Richie Bielak
Modula-2 Programming by Steve Faiwiszewski
The gameport device and simple sprites in action.
AmigaNotes by R. Rae A1000! Software-switchable output filter.
Roomers by The Bandito AmiExpo, Kickstart 1.4, Commodore
The Big Picture by Warren Ring-- Unified Field Theory!!
Plus a great collection of monthly columns...

Volume 3 Number 4 1988

Highlights from AmiExpo, Los Angeles by Steve Hull
Writing a SoundScape Patch Librarian T. Fay System Exclusive
Upgrade Your A1000 to A500/2000 Audio Power—by H.Bassen
Modifications to help your A1000 make sweet music, too!
Amiga Audio Guide Listing of all Amiga audio products.
Gels in Multi-Forth by John Bushakra
Macrobatics by Patrick J. Horgan Ease the trauma of assembly language programming.
Amiga Audio Sources The folks behind all those audio products.
Take Five! by Steve Hull five Amiga games reviewed.
Amiga Notes by Rick Rae A basic tour of Amiga audio.
The Ultimate Video Accesory, Part V by Larry White
Bug Bytes by John Steiner
The Big Picture by Warren Ring Part II Unified Field Theory.
Roomers by The Bandito Hardware hijinx,Toasted video, and more!
In the Public Domain by C.W. Flatte
Time Bandit review by Keith Conforti
AudioMaster review by B. Larson Real-time digitizing samples.
Music Mouse review by J Henry Lowengard
Making music without lifting a finger from the mouse.
Amiga-Tax/Canadian Version review by Ed Bercovitz
A Canadian income tax planning, preparation, & analysis package.
SAM BASIC review by Bryan Catley
A new BASIC which exploits even more unique Amiga features.

Volume 3 Number 5 1988

Interactive Startup Sequence by Udo Pernisz
The Command Line part 1by Rich Falconburg
AmigaTrix III by Warren Block—Tips and tidbits to ease Amiga life
Amiga Product Guide: Hardware Edition
Proletariat Programming by P.Quaid—Public domain compilers
The Companion by P.Gosselin Amiga's Event Handling capability.
MindLight 7 reviewed by David N. Blank
VideoScape 3-D 2.0 reviewed by David Hopkins
Extend reviewed by Bryan D. Catley—An AmigaBASIC extension
AssemPro reviewed by S.Kemp Opening assembly langua
APL.68000 reviewed by Roger Nelson
Book Reviews by Richard Grace—Three "C" programming texts.
CBTREE reviewed by Michael Listman C programmer. aid
The Big Picture by Warren Ring 3 part Unified Field Theory ends
Modula-2 by S.Faiwiszewski Termination mods for Benchmark &TDI
68000 Assembly Language by Chris Martin display routines.
Plus a great collection of monthly columns...

Volume 3 Number 6 1988

Bear Time Reviewed by Steve Carter A1000 battery-backed clock
Acquisition Reviewed by D.N. Blank a powerful relational database.
Butcher 2.0 Reviewed by G.Hull diverse image processing utilities.
Reassigning Workbench Disks by John Kennan
Endless disk swapping comes to a merciful end.
Product Guide: Software Tools Edition put your Amiga to work.
An IFF Reader in Multi-Forth by Warren Block
Basic Directory Service Program by Bryan Catley
A programming alternative to the GimmeeZeroZero windows.
C Notes from the C Group by Stephen Kemp C programming intro.
An Amiga Forum Conference with Jim Mackraz
Son of Seven Assemblers Reviewed by Gerald Hull
The 1988 Commodore Amiga Developers Conference
A look inside the conferences held in Washington, D.C.
Amiga Working Groups by Perry Kivolowitz and Eric Lavitsky
An outline of the innovative Amiga Working Groups concept.
The Command Line by Rich Falconburg
Exploring the multi-talented LIST command.
Plus a great collection of monthly columns...

Volume 3 Number 7 1988

Look, Up On the Screen, it's an Ami... It's a Pro... It's SuperGen reviewed by Larry White—Genlock comparisons
An Interview with "Anim Man," Gary Bonham by B. Larson
An animated conversation with the man behind the format.
The Amiga at Spring COMDEX in Atlanta by Ed Bercovitz
Amiga Product Guide: Video/Graphics Edition
Thirteen pages devoted to the Amiga's dazzling strong suit.
The Developing Amiga by Steve Pietrowicz Developers' notes
Roll Those Presses! by Barney Schwartz
Welcome to the dandy, demanding world of desktop publishing!
Linked Lists in C by W. E. Gammill Put dynamic memory to work!
FrameGrabber Preview by Oran Sands
Capturing an image can now be as fast as punching a single key!
A First Look at Interchange reviewed by David Hopkins
Bridge the gap between those incompatible animation packages.
Perfect Vision reviewed by Bryan Catley
Capture, digitize and save pictures from any video source.
ProWrite 2.0 Review reviewed by Pamela Rothman
A graphic word processor specializing in efficient editing.
Doug's Math Aquarium: The Art of Mathematics by R. Bielak
Bear Products MegaRex II Expansion RAM by Steve Carte
The Command Line by Rich Falconburg
Amiga Notes by Rick Rae The Other Guys' Synthia digital synthesizer
C Notes from the C Group by Stephen Kemp
Weathering the unknown "C" of basic object and data types.
Plus a great collection of monthly columns...

Volume 3 Number 8 1988

The Command Line by Rich Falconburg CLI linstruction
The Developing Amiga by Stephen R. Pietrowicz
A gaggle of great programming tools.
Modula-2 Programming by Steve Faiwiszewski
Libraries and the FFP and IEE Math Routines.
C Notes from the C Group by Stephen Kemp Arrays and pointers
Dark Castle reviewed by Keith Conforti—The Black Knight lurks
Ports of Call reviewed by Julie Landry
Leathernecк reviewed by Michael Creeden-Rambo's not so tough!
Capone reviewed by Joyce and Robby Hicks- Light Guns blaze
Casino Fever reviewed by Michael T. Cabral—Vegas on Amiga
Ferrari reviewed by Jeffery Scott Hall—Start your engine
Arkanoid reviewed by Graham Kinsey—"blockbuster"
Ebonstar by Keith Conforti —black hole treking.
Deluxe Productionsreviewed by Harv Laser—Video wizardry
Game Pizazz by Jeffery Scott Hall— Register your questions here.
TrackMouse by Darryl Joyce
Convert a standard Atari trackball into a peppy Amiga TrackMouse.
Amiga Interface for Blind Users reviewed by Carl W. Mann
An ingenious interface that opens the Amiga to even more users!
Video in the Sunshine State reviewed by Stephen R. Pietrowicz
RGB Video Creations hosts a video unveiling!
Amiga Product Guide: Games Edition
Tumblin' Tots by David Ashley—assemply language program.
Plus a great collection of monthly columns...

Volume 3 Number 9 1988

The Kideo Tapes by John Dandurand
A Georgia elementary school puts desktop video to work.
Speeding Up Your System by Tony Preston floppy disk caching
Amiga Product Guide: Education Edition
Everything you need to send your Amiga to the head of the class.
Computer Aided Instruction by P.Castonguay lin AmigaBASIC.
Gels in Multi-Forth, Part II: Screenplay by John Bushakra
Make the IFF converter from Part I easy to use-gadgets, menus,etc.
AmiExpo Midwest '88 by Michael T. Cabral Amiga wows Chicago
Intellitype by Harv Laser—Learning to type made easy...and *fun*?
Shakespeare by Barney Schwartz—Desktop publishing in full color.
XSpecs 3D by Steve Hull—A new dimension in Amiga graphics.
AmigaNotes by Richard Rae—How IFF sound samples are stored?
Take Five! by Steve Hull—Beat the back-to-school blues !
The Command Line by Rich Falconburg—continuing tour of CLI.
C Notes from the C Group by Stephen Kemp
Operators, expressions, and statements in C uncovered.
Roomers by The Bandito Can Apple IIgs Plus keep Amiga away?

Volume 3 Number 10 1988

A First Look At Deluxe PhotoLab reviewed by David Duberman
DiskMaster reviewed by Steve Hull —file management utility.
DSM: A MC680000 Disassembler reviewed by Gerald Hull
Looking for easily modifiable, assembler-ready code?
FBasic Language System reviewed by Patrick Quaid
BASIC compiler and development system.
Hot on the Shelves by Michael T. Cabral—Deviant dice, gripping gray scales, color cartography, mauling modems, and much more.
The Command Line by Rich Falconburg
NEWCLI: A painless way to create a new console window.
The Developing Amiga by S. Pietrowicz Usenet—24-Hour News
C Notes from the C Group by Stephen Kemp —loops
Roomers by The Bandito WP wars, ignominious interfaces, & more
PD Serendipity by C.W. Flatte—Fred Fish collectionpasses 150.
Comparison of MultiScan Monitors by Steven Bender
Record Keeping for Free-lancers:
A Superbase Professional Tutorial by Marion Deland
Record keeping system for free-lance photographers and others.
On The Crafting of Programs by David J. Hankins—A look at optimization kicks off a series of articles on programming savvy.
Bob and Ray Meet Frankenstein by Robert D'Asto—Create, animate, and metamorphose graphics objects in AmigaBASIC.
Digital Signal Processing in AmigaBASIC by Robert Ellis
Perform your own digital experiments with Fast Fourier Transforms.
HAM & AmigaBASIC by Bryan Catley—Pack your AmigaBASIC programs with many of the Amiga's 4096 shades!
CAI—Computer Aided Instruction: Part II by Paul Castonguay
The Editor program wraps up our authoring system in AmigaBASIC.

Volume 3 Number 11 1988

Desktop Publishing with Professional Page by Barney Schwartz tutorial in document creation, plus some jazzy enhancements.
Game Pizzazz by J. Hall gaming hints, tips, high-score secrets.
Structures In C by Paul Castonguay C programming in an nutshell.
On The Crafting of Programs by D. Hankins speed up your progs.
Desktop Video VI: Adding the Third Dimension by Larry White
Unraveling the complexity of 3D for your video creations.
A2000 Hard Drive Round Up by Sheldon Leemon
Keyclick by Mike M. Duppong a typewriter click in your keyboard.
More Linked Lists In C: Techniques and Applications by Forest W. Arnold Procedures for managing lists, storing diverse data types in the same list, and putting lists to work in your programs.
BASIC Linker by Brian Zupke Combine individual routines from your program library to create an executable program.
The Developing Amiga by Steven Pietrowicz
A look at mysteries and successes behind efficient beta testing.
Modeler 3D Preview reviewed by David Hopkins
A peek inside a new, open-ended 3D package.
APro Draw Graphics Tablet reviewed by Keith Conforti
Artists! Meet the future of Amiga graphics.
StarGlider II reviewed by Jeffery Scott Hall
Those irritating Ergons are back for another laser-lashing.
WShell reviewed by Lawrence Lichtman CLI substitute,
Hot on the Shelves by M. Cabral viruses, music, microfiche mastery
PD Serendipity by C.W. Flatte Fred Fish disks 149-152.
Roomers by The Bandito Golden RAM, 16-bit videogames, CD-I, another HAM skirmish ... what could possibly be NeXT?

Volume 3 Number 12 1988

Hot on the shelves by M. T. Cabral
Graphic adventure, control over Preferences, a Postscript print utility, sequence live action animation, a new deal for user groups and the figure construction set.
PD Serendipity by C. W. Flatte Fred Fish disks 158-162
Roomersby The Bandito
AmiExpo, C.D. the latest from Commodore and more.
AmiExpo California By Stephen Kemp Hot— All the news.
EMPIRE reviewed by Stephen Kemp
EMPIRE , the game of conquest, has finally come to the Amiga.
Virus Infection Protection (V.I.P.) reviewed by Jeffery Scott Hall
What makes a computer sick and the cure.
The Command Line by Rich Falconburg
What to do when the commands of AmigaDos fail.
Converting Patch Librarian Files by Phil Saunders
How to get your sounds from there to here.
E.C.T. SampleWare by Tim Mohansingh
The E.C.T. samples contain several gems.
The Creation of Don Bluth's Dragon's Lair by Randy Linden.
Easy Menus in JForth by Phil Burk HELLO WORLD.
Extending AmigaBasic by John Kennan
The use of library calls from within AmigaBASIC.
Better Dead Than Alien reviewed by Jeffery Scott Hall
Don't fire until you see the greens of their eyes.
Getting Started In Assembly by Jeff Glatt
An introduction to Amiga assembly language programming
AC/BASIC 1.3 reviewed by Bryan Catley
Release 1.3 of Absoft's AC/BASIC compiler for the Amiga.
Thexder reviewed by Bruce Jordan
Action, Adventure, Fantastic Sound, and stunning Graphics.
Magellan: The AMIGA Gets Smart reviewed by Steve Gillmor
Artificial intelligence comes to the AMIGA
C Notes From The C Group by Stephen Kemp
Program or function control coding: the case history.
AmigaDos, Assembly Language, And FileNotes by Dan Huth
Help against file overload; accurate, descriptive file naming.

Volume 4 Number 1 1989

The Wonderful World of Hashnique reviewed by Shamms Mortier
A review of the Amiga software products of Hash Enterprises
Desktop Video by Richard Starr
Thinking about getting into Video? Here's what you'll need to know.
Industrial Strength Menus by Robert D'Asto
Add some snazzy submenus to your AmigaBASIC cuisine
Second Generation 2D Animation Software by Geoffrey Williams
Cel Animators and Key Frame Animators,
how they differ and a look into their use.
What's The Diff? reviewed by Gerald Hull
A review of Lattice's Compiler Companion
Scrolling Through SuperBitMap Windows by Read Predmore
Implement SuperBitMaps for viewing/drawing into large graphic areas.
Alive in 3D by Shamms Mortier
A review of Caligari, a High-End 3D sculpting & animation package.
Sync Tips by Oran J. Sands III
Dot crawl, the Amiga and composite video devices.
How May I Animate Thee?, Let Me Count The Ways- by Shamms Mortier
An overview of animation techniques.
Stop-Motion Animation On The Amiga by Brian Zupke
A hands on approach to animation and the Amiga.
Roomers by The Bandito
Commodore's deal, RAM chip crisis, and more!
C Notes From the C Group by Stephen Kemp
Structures - A powerful feature of C
On the Crafting of Programs by David J. Hankins
What Format is right for you
The Command Line by Rich Falconburg
A look at new and improved Assembly Language commands
Questron II reviewed by Jeffery Scott Hall
Questron II - It's a journey back in time
Pointers, Function Pointers, and Pointer Declarations in C by Forest W. Arnold
Reducing data type dependencies
Las Vegas Comdex Report by Louise Brinkmann
Commodore's new 2500, 2500 UX, and more!
Philadelphia World of Commodore by Chris Darsch & Rick Rae
Highlights of Philadelphia's Commodore Show
STELLARYX Review by Stephen Kemp
Exciting & challenging! Terrific stereo and sound effects
Arkanoid Imposters: Unmasking the Impostors. reviewed by Jeffery Scott Hall
A look at Arkanoid look-alikes
Bug Bytes by John Steiner
Bugs and upgrades
Death of a Process by Mark Cashman
Develop an error handling module in Modula-2

To be continued.....

FULLY UTILIZING THE 68881 MATH COPROCESSOR

Turbo Mandelbrot and Julia Set Calculations PART II

by Read Predmore

Introduction

This is Part Two of a series of articles on assembly language programming of the Motorola MC68881 math coprocessor. Part One in the March 1989 issue of *Amazing Computing* [Ref. 1] discussed the speeding up of the Savage benchmark. This month I will discuss something with more visual appeal—calculating Mandelbrot and Julia sets.

First I will give a brief overview of Mandelbrot and Julia sets and then discuss a common algorithm which can calculate either one. I will describe several example Mandelbrot and Julia sets. Finally, I'll describe various program files which are used to display these sets.

I want to emphasize that the MC68881 macros and the assembler source code in these articles are not specific to the MicroBotics Starboard 2 unit. Other implementations of the MC68881 math chip will run this code by changing only the base address of the '881 chip. However, the routines as written are not multitasking. A multitasking version would require rewriting the Amiga task scheduling handlers to save and restore the state of the MC68881 chip.

Calculating Mandelbrot and Julia sets

The Mandelbrot and Julia sets are calculated according to algorithms, which are described in *The Beauty of Fractals* by Peitgen and Richter [Ref. 2]. Although there are numerous books available on Mandelbrot sets and fractals, I am using *The Beauty of Fractals* since the algorithm is symmetric enough to be used for both Mandelbrot and Julia sets, and they have lovely sample pictures as well.

Both Mandelbrot and Julia sets are calculated over rectangular areas in the complex plane. However, you do not have to know a thing about complex numbers to follow these calculations, since each complex number is represented by a point in the X-Y plane. For example, the complex number Z is represented by the pair of numbers (Zx, Zy). The Mandelbrot or Julia sets are calculated over a rectangular areas whose corners are given by (Xmin, Ymin) and (Xmax, Ymax).

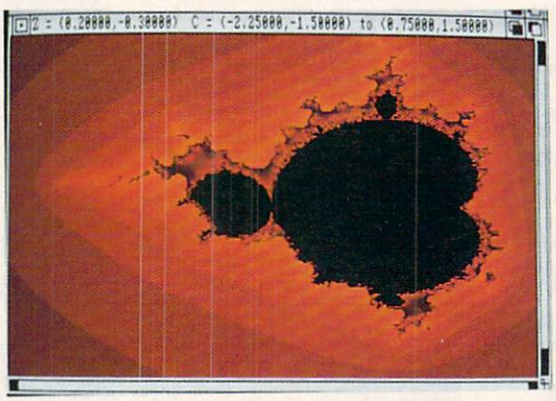
For a Mandelbrot set, the X-Y plane represents the complex number C. For Julia sets, the X-Y plane represents the complex number Z. "So what?", you might ask (with justification). In a mathematical sense, the Mandelbrot and Julia sets are perpendicular or orthogonal to each other. There is a Julia set associated with each point of the Mandelbrot set. This is illustrated in a schematic way in Figure One, which has the Mandelbrot set in perspective with parts of Julia sets sticking out like playing cards. Although this figure is not strictly true in a mathematical sense, it does represent the orthogonality of the Mandelbrot and Julia sets.

The axes Cx, Cy, Zx and Zy actually define a four-dimensional hyperspace, where the Mandelbrot and Julia sets are only two of the possible six planes which can be viewed. The Mandelbrot set is represented by the Cx-Cy plane for a given point (Zx, Zy), and a Julia set is in the Zx-Zy plane for a given point (Cx, Cy). The other possible planes and their associated points are:

Plane	Point
Cx-Zx	(Cy, Zy)
Cx-Zy	(Cy, Zx)
Cy-Zx	(Cx, Zy)
Cy-Zy	(Cx, Zx)

I just mention these other possibilities since most articles do not discuss them and the extension to the MANDEL_881 program to include them is straightforward.

Figure One *(right)*
A Mandelbrot set in perspective with parts of Julia sets sticking out like playing cards.



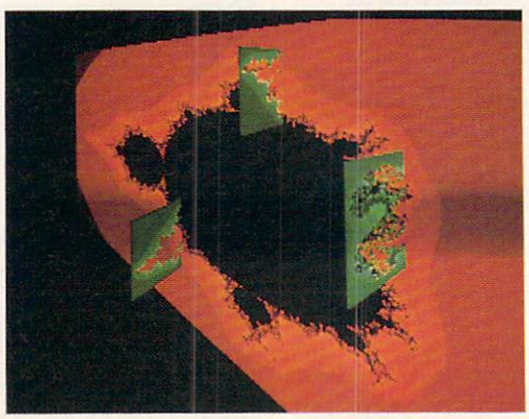


Figure Two *(left)*
interesting non-symmetrical Mandelbrot sets can be generated from non-zero values for cx_in and cy_in.

The function calc_dble() uses X, Y coordinates, which define a box over which either the Mandelbrot or Julia set is to be calculated. This function calculates the limits for one row of pixels at a time. Between rows, the program can respond to user interactions such as resizing or scrolling the SuperBitMap window, or selecting the Close gadget so the program can be stopped.

Once the parameters are set up for a point, the calculations are the same for both the Mandelbrot and Julia sets. The fascinating patterns generated are found by repeatedly squaring a number and comparing the squared number with the maximum squared number. More information is given below in the section on details of calculations.

Using the programs

The program MANDEL_881 is used with a one-line text file with several parameters. Table One has parameters for four Julia sets and seven Mandelbrot sets. The first parameter is a string of characters starting with either JULIA or MANDE in capital letters. In the table, the _18, etc. stand for maps in the book *The Beauty of Fractals*. It is important that there be no blank spaces in this label or the second group of characters will be interpreted as a number. Following the label are seven floating point numbers representing Xmin, Xmax, Ymin, Ymax, Cx_in, Cy_in and Z2max. The last number is a positive integer which is the initial value for the maximum number of iterations. If the program is left running, the maximum number of iterations is doubled after each complete pass through the set.

Details of calculations

Listing Two gives the calc_dble() function which calculates either the Mandelbrot or Julia sets for a row of pixels. The starting and ending pixel numbers are nx_min and (nx_max-1).

The maximum number of iterations is given by max_iter, and the number of iterations found for each point are stored in the results[] vector. The criteria for stopping the iteration for each pixel is that the magnitude squared of z, which is the variable z2, is greater than or equal to the input variable z2max.

For each pixel, the local variables cx, cy, zx and zy are initialized using the input variables cx_in, cy_in, xmin and x. For Mandelbrot sets, zx and zy are constants for the entire row and are set to the variables cx_in and cy_in, respectively. For the usual Mandelbrot set, cx_in and cy_in are both equal to zero, but other interesting non- symmetrical Mandelbrot sets can be generated from non-zero values for cx_in and cy_in. An example is shown in Figure Two for Z=(0.2,-0.3) over C=(-2.25,-1.5) to (-1.5,1.5). The current value of cx for pixel nx is given by cx = xmin + nx*dx.The variable cy is set to y_in, which is the Y value of the current row.

For Julia sets, which are calculated for rows of points in the Zx-Zy plane, the initialization is:

```
cx = cx_in;
cy = cy_in;
zx = xmin + nx*dx;
zy = y_in;
```

Once the C and Z variables are initialized for either a Mandelbrot or Julia set, the iteration procedure is the same:

```
for(num_iter=0; num_iter<max_iter; num_iter++)
{  zy2 = zy*zy;
   z2 = zx*zx + zy2;
   if(z2 >= z2max)
      break;
   zy = 2.*zx*zy + cy;
   zx = zx*zx - zy2 + cx;
}
results(nx) = num_iter;
```

The break command causes a branch out of the loop where the number of iterations is stored in the results[] vector. The calculations within the loop are further broken down into single operations in Listing Two so the code can be converted into '881 macros more easily.

Converting to '881 assembler

To utilize the '881 macros in file MC68881.I of Part I [Ref. 1], the Manx Amiga C compiler is used to compile the file CALC_DBLE.C with the command:

```
CC +FI -A -N -T -O CALC_881.ASM CALC_DBLE.C
```

which gives an assembly listing with the file name CALC_881.ASM [Listing Three]. The -T option is used to keep the original C listing as comments. The -N option gives debug-

(continued)

Table One
Sample Mandelbrot and Julia Set Parameters

Type	XMIN	XMAX	YMIN	YMAX	CX_IN	CY_IN	Z2MAX	Num Iterations
JULIA_18	-2.00	2.00	-1.50	1.50	0.32000	0.04300	100.0	64
JULIA_20	-2.00	2.00	-1.50	1.50	-0.12375	0.56508	100.0	16
JULIA_22	-1.50	1.50	-1.50	1.50	-0.39054	-0.58679	100.0	16
JULIA_24	-2.00	2.00	-1.50	1.50	-0.11000	0.67000	100.0	16
MANDEL_26	-2.25000	0.75000	-1.50000	1.50000	0.0	0.0	100.	16
MANDEL_27	-0.19920	-0.12954	1.01480	1.06707	0.0	0.0	100.	32
MANDEL_29	-0.95000	-0.88333	0.23330	0.30000	0.0	0.0	100.	32
MANDEL_30	-0.71300	-0.40820	0.49216	0.71429	0.0	0.0	100.	32
MANDEL_33	-1.78100	-1.76400	0.00000	0.01300	0.0	0.0	100.	32
MANDEL_36	-0.75104	-0.74080	0.10511	0.11536	0.0	0.0	100.	64
MANDEL_38	-0.74758	-0.74624	0.10671	0.10779	0.0	0.0	100.	128

ging information for the Manx SDB source level debugger. I had hoped to include a debugging function which printed out the '881 floating point registers in this article, but time constraints have postponed that to the next article.

The lines in CALC_881.ASM, which start with ~ or ^, provide information for the Manx SDB debugger. For example, input variables such as cx_in have a positive offset (24) with respect to the stack pointer register A5 of the 68000 CPU. And local variables such as cx have a negative offset (- 40) with respect to A5. The variable nx is a register variable in the D5 data register of the 68000 CPU.

Numerous modifications or additions are now made to the original CALC_881.ASM file. After the include file MC68881.I is read in, seven variables, zy2, zx, zy, cx, cy, z2max and z2 are EQUated to the last seven floating point registers (FP1 to FP7) internal to the '881. The EQUates between variable names and '881 registers dramatically improve the readability and ease of debugging of the program. Where code has been modified, the original code has been commented out with semicolons and moved to the right of the line. The new code is at the left part of the line.

Outside of the loop over nx, z2max is stored in the '881 by moving it to (D0/D1) and using the macro:

```
PROCD0D1toFPN fmove,z2max
```

At the start of the nx_loop, the variables cx, cy, zx and zy are initialized according to the mand_flag and stored into the '881 floating point registers.

Most of the '881 macros were discussed last time, except for the floating point comparison and branch macro. This is used on lines 185 and 186 of the file CALC_881.ASM to test if (z2 >= z2max):

```
REGREG   fcmp,z2max,z2
FBCC.s   ge,set_result
```

First, the floating point comparison (fcmp) in used on the registers containing z2max and z2. Then the test for greater-than-or-equal -to(ge) is used and, if TRUE, a branch to the label set_result is taken.

At the end of the assembler listing, the SHUTDOWN_881 macro is used to restore the A2 register. An addition to the data segment (dseg) part of the listing is the public reference to _MC68881_BASE. This is a global variable in the C program and needs to be referenced during the SETUP_881 macro. It contains the physical address of the MC68881 chip and is found by the test_881() function.

Plotting the results

The function plot_results.c is a separate listing [Four] since it will be modified when the turbo_pixel() function is discussed in the next article. The results[] vector contains the number of iterations for each point in the row of pixels given by NY. The first two colors are reserved for the border color except for color zero, which is used by points which have the maximum number of iterations. The numbers that have been calculated are cyclically mapped onto the remaining colors with the modulus function (%). The pixel color is then changed with the SetAPen() function and the pixel is actually written with the WritePixel() function. Normally, nx_min is 0 and nx_max is set to the SuperBitMap width, but the routine was written with the possibility of writing only part of a row.

Program listings

The source code listings MAND_881.C and MANDSUBS.C, which have been adapted from the SCROLL.C listing of Ref. 3, are given in Listings One and Six. The functions calc_dble() and calc_881() (Listings Two and Three) are the additions in the calculation of the Mandelbrot and Julia sets. In MAND_881.C, the test_881() function has been trimmed down by eliminating the print statements so the program can be run from an icon.

The input of the various parameters has been separated into the get_mdata() function so that eventually data could be entered via an ASCII file, a requestor or an ARexx port.

The MAKEFILE for this set of program files is included in Listing Seven.

The include file MANDEL.H [Listing Five] defines various screen and window parameters, as well as the SuperBitMap size.Numerous void functions are also defined. Finally, global variables are defined as external for files other than the main module MAND_881.C.

Corrections

Two errors in Part I [Ref. 1] have come to my attention. In Table Two, the SETUP_881 and SHUTDOWN_881 macros only utilize the A2 register of the 68000 CPU. The register A1 was used until I discovered there was a conflict with the printf() function.

More importantly for this month's program is a missing left parenthesis in the FBCC macro in Listing Four for the MC68881.i include file. On page 75, the first line after FBCC macro .... should read

```
move.w  #\1,condition(a2)
```

To be continued...

Next time I will finally provide the '881 debugging function, give timing results for various mathematical libraries and discuss the turbo_pixel() routine for changing the colors of a set of contiguous pixels.

References

1. "Fully Utilizing the Motorola 68881 Math Coprocessor: Part 1, Turbocharging the Savage Benchmark", R. Predmore, *Amazing Computing*, Vol. 4 No. 3, pp. 69-75, March 1989.

2. *The Beauty of Fractals*, H.O. Peitgen and P.H. Richter, Springer-Verlag, New York, 1986.

3. "Scrolling Through SuperBitMap Windows", R. Predmore, *Amazing Computing*, Vol. 4 No. 1, pp. 101-108, January 1989.

Listing One
MAND_881.C

```
/*====================< MAND_881.C >=======================
 *
 *  Copyright (C) 1989
 *  by PiM Publications & Read Predmore
 *  All Rights Reserved
 *  26 March 1989 @ 13:17
 */

#define  MAIN_MODULE 1
#include "mandel.h"

long     cur_resource;

/*================< GRAPHICS DEFINITIONS >================*/

struct  GfxBase       *GfxBase;
struct   IntuitionBase *IntuitionBase;
struct  LayersBase    *LayersBase;
UBYTE    screen_title[80], window_title0[80],
         window_title1[80];
struct  BitMap  *pSBM;
struct  Screen  *pscreen0;
struct  Window  *pwindow0;
struct  Image     Horiz_image, Vert_image;
struct  PropInfo Horiz_prop,  Vert_prop;

struct  Gadget  Horiz_gadget =
{   NULL,
    1, -7, -16,  8,
    GADGHCOMP | GRELBOTTOM | GRELWIDTH,
    RELVERIFY | GADGIMMEDIATE | FOLLOWMOUSE | BOTTOMBORDER,
    PROPGADGET | GZZGADGET,
    (APTR) &Horiz_image,
    NULL,
    NULL,
    0,
    (APTR) &Horiz_prop,
    HORIZ_SCROLL,
    NULL
};
struct  Gadget  Vert_gadget =
{   &Horiz_gadget,
    -15, 10,  16, -18,
    GADGHCOMP | GRELRIGHT | GRELHEIGHT,
    RELVERIFY | GADGIMMEDIATE | FOLLOWMOUSE | RIGHTBORDER,
    PROPGADGET | GZZGADGET,
    (APTR) &Vert_image,
    NULL,
    NULL,
    0,
    (APTR) &Vert_prop,
    VERT_SCROLL,
    NULL
};

/* Global argument count which will be zero if started
   from Workbench. */
   int  Gargc;

#define  JULIA_SET   0x0001
#define  MANDEL_SET  0x0002

struct    Mandel_Data
{   DOUBLE   xmin, xmax, ymin, ymax, cx, cy, z2max;
    unsigned short  max_iter;
    unsigned char   mflag;
};

void  *MC68881_BASE;

/*======================< MAIN >======================*/
main(argc, argv)
int     argc;
char      *argv[];
{
    DOUBLE   xmin, xmax, ymin, ymax, cy, dy, cx, dx, zy,
             z2max;
    long     dxmax, dymax;
    SHORT    keep_going = (SHORT) TRUE;
    time_t   start_time, stop_time;
    ULONG    gadget_state;
    unsigned short *results;
    unsigned short  gadget_id, max_iter,
                    nx_min, nx_max, ny, ny_min, ny_max;

    struct  IntuiMessage  myIntuiMessage, *tempIntuiMessage;
    struct  Window       *pcur_wind;

    struct  Mandel_Data   mdata, get_mdata();

    unsigned char mand_flag = 0;

    unsigned long i, total_iter = 0;

    cur_resource = 0;
    Gargc = argc;

    test_881();

    if(argc<2)
    {   if(Gargc)
            (void) printf("USAGE: %s input_filename\n",
                  argv[0]);
        myabort(" ");
    }
    mdata = get_mdata(argv[1]);
    xmin  = mdata.xmin;
    xmax  = mdata.xmax;
    ymin  = mdata.ymin;
    ymax  = mdata.ymax;
    cx    = mdata.cx;
    cy    = mdata.cy;
    z2max = mdata.z2max;
    max_iter = mdata.max_iter;
    mand_flag = mdata.mflag;

    if(!mand_flag)
       myabort("Neither MANDELBROT or JULIA set specified");

    initialize();
    pcur_wind = pwindow0;

    if(mand_flag == MANDEL_SET)
    {   sprintf(window_title1,
       "Z=(%.5f,%.5f)  C=(%.5f,%.5f) to (%.5f,%.5f)        "
              , cx, cy, xmin,ymin, xmax, ymax);
        SetWindowTitles(pcur_wind, window_title1, -1L);
    }
    else if(mand_flag == JULIA_SET)
    {   sprintf(window_title1,
       "C=(%.5f,%.5f)  Z=(%.5f,%.5f) to (%.5f,%.5f)        "
              , cx, cy, xmin,ymin, xmax, ymax);
        SetWindowTitles(pcur_wind, window_title1, -1L);
    }

    nx_min = 0;
    nx_max = (unsigned short)
```

(continued)

```
            (8*pcur_wind->WLayer->SuperBitMap->BytesPerRow);

    if(! (results = malloc((unsigned int) (sizeof
                            (unsigned long)*nx_max))) )
         myabort("Cannot allocate memory for results");

    dxmax = (long) ((8 *
                pcur_wind->WLayer->SuperBitMap->BytesPerRow)
                    - pcur_wind->GZZWidth);
    dymax =(long)
(pcur_wind->WLayer->SuperBitMap->Rows-pcur_wind->GZZHeight);
    gadget_state = GADGETUP;
    scroll_SBM(pcur_wind, &Horiz_prop, &Vert_prop, dxmax,
               dymax, (USHORT)(HORIZ_SCROLL | VERT_SCROLL));

    dx  = (xmax-xmin) / ((DOUBLE) (nx_max-1));
    ny_min = 0;
    ny = ny_min;
    ny_max = (unsigned short)
             pcur_wind->WLayer->SuperBitMap->Rows;
    dy  = (ymax-ymin) / ((DOUBLE) (ny_max-1));

    start_time = time((time_t *) NULL);
    while(keep_going)
    {
        zy = ymax - ny * dy;
        if(MC68881_BASE)
            calc_881(xmin, dx, cx, cy, zy, z2max, mand_flag,
                     results, max_iter, nx_min, nx_max);
        else
            calc_dble(xmin, dx, cx, cy, zy, z2max,mand_flag,
                      results, max_iter, nx_min, nx_max);
        plot_results(pcur_wind, results, max_iter, nx_min,
                     nx_max, ny);
     /*
          for(i=nx_min; i<nx_max; i++)
              total_iter += results[i];
     */
        ny++;
        if(ny == ny_max)
        {   stop_time = time((time_t *) NULL);
            ny = ny_min;
            max_iter *= 2;
            start_time = time((time_t *) NULL);
        }
        while(tempIntuiMessage = (struct IntuiMessage *)
              GetMsg(pcur_wind->UserPort))
        {   myIntuiMessage = *tempIntuiMessage;

            ReplyMsg(tempIntuiMessage);
            switch(myIntuiMessage.Class)
            {   case GADGETUP:
                    gadget_id = ((struct Gadget *)
                        myIntuiMessage.IAddress)->GadgetID;
                    scroll_SBM(pcur_wind, &Horiz_prop,
                               &Vert_prop,
                               dxmax, dymax, gadget_id);
                    gadget_state = GADGETUP;
                    break;
                case GADGETDOWN:
                    gadget_state = GADGETDOWN;
                    gadget_id = ((struct Gadget *)
                        myIntuiMessage.IAddress)->GadgetID;
                    break;
                case MOUSEMOVE:
                    if(gadget_state == GADGETDOWN)
                        scroll_SBM(pcur_wind, &Horiz_prop,
                                   &Vert_prop, dxmax, dymax,
                                   gadget_id);
                    break;
                case NEWSIZE:
                    dxmax = (long) (8 *
              pcur_wind->WLayer->SuperBitMap->BytesPerRow)
                            - pcur_wind->GZZWidth;
                    dymax = (long)
                      pcur_wind->WLayer->SuperBitMap->Rows -
                            pcur_wind->GZZHeight;
                    scroll_SBM(pcur_wind, &Horiz_prop,
                               &Vert_prop, dxmax, dymax,
                               (USHORT) (HORIZ_SCROLL |
                                         VERT_SCROLL));
                    update_scroll_gadgets(pcur_wind,
                                &Horiz_prop, &Vert_prop);
                    break;
                case CLOSEWINDOW:
                    while(tempIntuiMessage =
                            (struct IntuiMessage *)
                            GetMsg(pcur_wind->UserPort))
                        ReplyMsg(tempIntuiMessage);
                    keep_going = (SHORT) FALSE;
                    break;
                default:
                    break;
            } /* End of switch loop. */
        } /* End of while(IntuiMessage) loop. */
    } /* End of while(keep_going) loop. */
    free(results);
    myabort("End of Mandel '881 program.");
}

/*====================< GET_MDATA >=========================
 *  Get data for Mandelbrot or Julia set calculations
 *  from requestor or file.
 *  Read Predmore, 13 September 1988.
 */
struct    Mandel_Data
get_mdata(filename)
char      *filename;
{
    char buffer[128];
    FILE *infp;
    struct    Mandel_Data    md;

    md.mflag = NULL;
    if((infp = fopen(filename, "r")))
    {
        (void) fscanf(infp, "%s", buffer);
        if( (strncmp(buffer, "MANDE", 5)) == 0)
            md.mflag = MANDEL_SET;
        else if( (strncmp(buffer, "JULIA", 5)) == 0)
            md.mflag = JULIA_SET;
        (void) fscanf(infp, " %lf", &md.xmin);
        (void) fscanf(infp, " %lf", &md.xmax);
        (void) fscanf(infp, " %lf", &md.ymin);
        (void) fscanf(infp, " %lf", &md.ymax);
        (void) fscanf(infp, " %lf", &md.cx);
        (void) fscanf(infp, " %lf", &md.cy);
        (void) fscanf(infp, " %lf", &md.z2max);
        (void) fscanf(infp, " %d",  &md.max_iter);
    }
    return md;
}
/*====================< INIT_COLOR_MAP >====================*/
/*
 *  Set up colors for the screen.
 */
void
init_color_map(pscreen)
struct  Screen  *pscreen;
{
    static short map_values[16] = {
/*Format 0x0RGB        0x0RGB         0x0RGB         0x0RGB */
/*  0 */ 0x0000, /* 1*/0x0BBB, /* 2*/0x0200, /* 3*/0x0300,
/*  4 */ 0x0400, /* 5*/0x0500, /* 6*/0x0600, /* 7*/0x0700,
/*  8 */ 0x0800, /* 9*/0x0900, /*10*/0x0A00, /*11*/0x0B00,
/* 12 */ 0x0C00, /*13*/0x0D00, /*14*/0x0E00, /*15*/0x0F00
    };
    LoadRGB4(&pscreen->ViewPort, map_values, 16L);
}
/*======================< TEST_881 >========================
 *  io68881.library test program and math chip locator
 *  by Jim Goodnow II */
struct {
    struct Library  io8_lib;
    void           *io8_68881;
} *lib_881;
void
test_881()
{
    lib_881 = (void *) OpenLibrary("io68881.library", 0L);
    MC68881_BASE = (void *) 0L;
    if(lib_881)
    {   MC68881_BASE = lib_881->io8_68881;
        CloseLibrary((struct Library *) lib_881);
    }
}
/*======================< FINI >==========================*/
```

Listing Two
Calc_dble.c

```
/*====================< CALC_DBLE.C >=======================
 *
 *  Copyright (C) 1989
 *  by PiM Publications & Read Predmore
 *  All Rights Reserved
 *  27 March 1989 @ 12:07
 */

#define  JULIA_SET   0x0001
#define  MANDEL_SET  0x0002

/*====================< CALC_DBLE >=====================*/
void
calc_dble(xmin, dx, cx_in, cy_in, y_in, z2max, mand_flag,
          results, max_iter, nx_min, nx_max)
double    xmin, dx, cx_in, cy_in, y_in, z2max;
unsigned  char   mand_flag;
unsigned  short *results, max_iter, nx_min, nx_max;
```

```
{
    register  unsigned short num_iter, nx;
    double    zx, zy, z2, zy2, cx, cy;

    if((mand_flag==MANDEL_SET) || (mand_flag==JULIA_SET))
        for(nx=nx_min; nx<nx_max; nx++)
        {
            if(mand_flag == MANDEL_SET)
            {   cx = xmin + nx*dx;
                cy = y_in;
                zx = cx_in;
                zy = cy_in;
            }
            if(mand_flag ==  JULIA_SET)
            {   cx = cx_in;
                cy = cy_in;
                zx = xmin + nx*dx;
                zy = y_in;
            }
            for(num_iter=0; num_iter<max_iter; num_iter++)
            {   zy2 = zy*zy;
                zy *= zx;
                zx *= zx;
              /*    z2 = zx*zx + zy*zy        */
                z2 = zx + zy2;
                if(z2 >= z2max)
                    break;
            /*    zy = 2.*zx*zy + cy;       */
                zy += zy;
                zy += cy;
            /*    zx = zx*zx - zy*zy + cx; */
                zx -= zy2;
                zx += cx;
            }
            results[nx] = num_iter;
        }
}
```

Listing Three
Calc_881.asm

```
;/*====================< CALC_881.ASM >=====================
; *
; *  Copyright (C) 1989
  *  by PiM Publications & Read Predmore
  *  All Rights Reserved
; *  28 March 1989 @ 15:28
; */
;
;#define    JULIA_SET    0x0001
;#define    MANDEL_SET   0x0002
;
;/*====================< CALC_881 >======================*/
;void
;calc_881(xmin, dx, cx_in, cy_in, y_in, z2max, mand_flag,
;          results, max_iter, nx_min, nx_max)
;double   xmin, dx, cx_in, cy_in, y_in, z2max;
# 14 'calc_dble.c' 354542835
^^| .3
    public    _calc_881
_calc_881:
    link      a5,#.2
    movem.l   .4,-(sp)
;unsigned char    mand_flag;
;unsigned short  *results, max_iter, nx_min, nx_max;
;{
;    register  unsigned short num_iter, nx;
;    double    zx, zy, z2, zy2;
;
~ cy  -48 "d"
~ cx  -40 "d"
~ zy2 -32 "d"
~ z2 -24 "d"
~ zy -16 "d"
~ zx -8 "d"
~ nx d5 "I"
~ num_iter d4 "I"
~~ xmin 8 "d"
~~ dx 16 "d"
~~ cx_in 24 "d"
~~ cy_in 32 "d"
~~ y_in 40 "d"
~~ z2max 48 "d"
~~ mand_flag 57 "C"
~~ results 58 "#I"
~~ max_iter 62 "I"
~~ nx_min 64 "I"
~~ nx_max 66 "I"

    include    MC68881.i

;Variable    '881 register
zy2       equ         fp1
zx        equ         fp2
zy        equ         fp3
cx        equ         fp4
cy        equ         fp5
z2max     equ         fp6
z2        equ         fp7

    SETUP_881                    ;  Setup for MC68881 math

;    if((mand_flag==MANDEL_SET) || (mand_flag==JULIA_SET))
^^^^^^^;         for(nx=nx_min; nx<nx_max; nx++)
    cmp.b  #2,57(a5)
    beq    .6
    cmp.b  #1,57(a5)
    bne    .5
.6
^    move.w    64(a5),d5

;   move z2max to z2max register in MC68881
    move.l    48(a5),d0
    move.l    52(a5),d1
    PROCD0D1toFPN    fmove,z2max

    bra    .10
nx_loop
;        {
^;            if(mand_flag == MANDEL_SET)
^;            {   cx = xmin + nx*dx;
    cmp.b  #2,57(a5)
    bne    .11
^    move.l  #0,d0
    move.w   d5,d0
    jsr      .Pflt#
    move.l   16(a5),d2
    move.l   20(a5),d3
    jsr      .Pmul#
    move.l   8(a5),d2
    move.l   12(a5),d3
    jsr      .Padd#
                                 ; move.l    d0,-40(a5)
    PROCD0D1toFPN    fmove,cx    ; move.l    d1,-36(a5)
;                cy = y_in;
^   move.l  40(a5),d0            ; move.l    40(a5),-48(a5)
    move.l  44(a5),d1            ; move.l    44(a5),-44(a5)
    PROCD0D1toFPN    fmove,cy
;                zx = cx_in;
```

(continued)

Amazing COMPUTING™ AMIGA
Your Original AMIGA® Monthly Resource

Volume 4 Number 5

Amazing Features
The Business of Video *by Steve Gillmor*
Get started in the video business.
An Amiga Adventure *by Larry White*
The globetrotting Amiga in Cologne, Germany.
Uninterruptible Power Supply (UPS), Part I *by Steve Bender*
Voltage spikes, surges, power failures? Are they uncommon?

Amazing Programming
The Amazing Audio Digitizer *by Andre Theberge*
Quality Amiga audio for less—building your own stereo digitizer.
A MIDI Out interface *by Br. Seraphim Winslow*
Helpful tips for happy jamming.
Digitized Sounds in Modula-2 *by Len A. White*
Produce impressive sound effects with sampled sounds.
Sync Tips *by Oran J. Sands*
The secrets hidden beneath the flicker mode.
On the Crafting of Programs *by David J. Hankins*
See how Lattice C 5.02 measures up.
Insta Sound in AmigaBASIC *by Greg Stringfellow*
The sounds you want for your program—in an instant!
Who are you, Mr. Guru? *by David Martin*
David exposes this Amiga deviant for what he really is.

Amazing Reviews
Gold Disk's Professional Draw *by R. Shamms Mortier*
The latest in professional drawing tools from Gold Disk.
Electronic Arts' DeluxePaint III *by David Duberman*
DPaint's paintbrush grows feet—combines paint with animation.
Aegis's AudioMaster II *by Phil Saunders*
Aegis's newest rendition of sound sampling & editing is reviewed.
New Wave Software's Dynamic Studio *by Chuck Raudonis*
New Wave's on a roll with this follow-up to Dynamic Drums.
Dr. T's MIDI Recording Studio *by Tim Mohansingh*
A high-performance, low-budget remedy for your MIDI ills.
Snapshot *by R. Brad Andrews*
Alien Syndrome and Tetris are among the new Amiga games.

Amazing Columns
New Products and Other Neat Stuff *by Michael Creeden*
Central Coast Software calls huddle over phony Qback 3.0, daVinci meets Disney with DPaint III, Blue Ribbon Bakery serves up organization, plus more.
Bug Bytes *by John Steiner*
VirusX 3.3 an evil twin, some bickering from Nag Plus 3.0, plus.
Roomers *by the Bandito*
The Bandito stalks AmiEXPO NY, Atari/Nintendo lawsuit expands, and the Beatles get a little help from their lawyers.
PD Serendipity *by C.W. Flatte*
C.W. covers Fred Fish from 189-200.
C Notes from the C Group *by Stephen Kemp*
Formatted output functions.

Index of Amazing Advertisers

Discover something interesting? Need more information?

Use the Reader Service Card in every issue of AC to contact those advertisers who have sparked your interest. Advertisers want to hear from you. This is the best way they have of determining the Amiga community's interests and needs. Take a moment and contact the companies with products you want to know more about. And, if you wish to contact an Amazing Advertiser directly, please tell them you saw their advertisement in Amazing Computing.

```
^   move.l  24(a5),d0           ; move.l  24(a5),-8(a5)
    move.l  28(a5),d1           ; move.l  28(a5),-4(a5)
    PROCD0D1toFPN    fmove,zx

;                zy = cy_in;
^   move.l  32(a5),d0           ; move.l  32(a5),-16(a5)
    move.l  36(a5),d1           ; move.l  36(a5),-12(a5)
    PROCD0D1toFPN    fmove,zy
;           }
^;           if(mand_flag ==  JULIA_SET)
.11
^;           {   cx = cx_in;
    cmp.b    #1,57(a5)
    bne      .12
^   move.l  24(a5),d0           ;  move.l  24(a5),-40(a5)
    move.l  28(a5),d1           ;  move.l  28(a5),-36(a5)
    PROCD0D1toFPN    fmove,cx
;           cy = cy_in;
^   move.l  32(a5),d0           ;  move.l  32(a5),-48(a5)
    move.l  36(a5),d1           ;  move.l  26(a5),-44(a5)
    PROCD0D1toFPN    fmove,cy
;               zx = xmin + nx*dx;
^   move.l   #0,d0
    move.w   d5,d0
    jsr      .Pflt#
    move.l   16(a5),d2
    move.l   20(a5),d3
    jsr      .Pmul#
    move.l   8(a5),d2
    move.l   12(a5),d3
    jsr      .Padd#
                                ; move.l  d0,-8(a5)
    PROCD0D1toFPN    fmove,zx   ; move.l  d1,-4(a5)
;                zy = y_in;
^
    move.l  40(a5),d0           ; move.l  40(a5),-16(a5)
    move.l  44(a5),d1           ; move.l  44(a5),-12(a5)
    PROCD0D1toFPN    fmove,zy

;           }
^;           for(num_iter=0; num_iter<max_iter; num_iter++)
.12
^   move.l   #0,d4
    bra      .16
iter_loop
;           {   zy2 = zy*zy;
^
                                ; move.l  -16(a5),d2
                                ; move.l  -12(a5),d3
                                ; move.l  -16(a5),d0
    REGREG   fmove,zy,zy2       ; move.l  -12(a5),d1
    REGREG   fmul,zy,zy2        ; jsr     .Pmul#
                                ; move.l  d0,-32(a5)
                                ; move.l  d1,-28(a5)
;               zy *= zx;
^
                                ; move.l  -8(a5),d2
                                ; move.l  -4(a5),d3
                                ; move.l  -16(a5),d0
                                ; move.l  -12(a5),d1
    REGREG   fmul,zx,zy         ; jsr     .Pmul#
                                ; move.l  d0,-16(a5)
                                ; move.l  d1,-12(a5)
;               zx *= zx;
^
                                ; move.l  -8(a5),d2
                                ; move.l  -4(a5),d3
                                ; move.l  -8(a5),d0
                                ; move.l  -4(a5),d1
    REGREG   fmul,zx,zx         ; jsr     .Pmul#
                                ; move.l  d0,-8(a5)
                                ; move.l  d1,-4(a5)
;               z2 = zx + zy2;
^
                                ; move.l  -32(a5),d2
                                ; move.l  -28(a5),d3
                                ; move.l  -8(a5),d0
    REGREG   fmove,zx,z2        ; move.l  -4(a5),d1
    REGREG   fadd,zy2,z2        ; jsr     .Padd#
                                ; move.l  d0,-24(a5)
                                ; move.l  d1,-20(a5)
;               if(z2 >= z2max)
^;                   break;
                                ; move.l  48(a5),d2
                                ; move.l  52(a5),d3
                                ; move.l  -24(a5),d0
                                ; move.l  -20(a5),d1
    REGREG   fcmp,z2max,z2      ; jsr     .Pcmp#
    FBCC.s   ge,set_result      ; bge     set_result
;/*              zy = 2.*zx*zy + cy;      */
;               zy += zy;
^^^
                                ; move.l  -16(a5),d2
                                ; move.l  -12(a5),d3
                                ; move.l  -16(a5),d0
                                ; move.l  -12(a5),d1
```



```
    REGREG   fadd,zy,zy         ; jsr     .Padd#
                                ; move.l  d0,-16(a5)
                                ; move.l  d1,-12(a5)
;               zy += cy;
^
                                ; move.l  -48(a5),d2
                                ; move.l  -44(a5),d3
                                ; move.l  -16(a5),d0
                                ; move.l  -12(a5),d1
    REGREG   fadd,cy,zy         ; jsr     .Padd#
                                ; move.l  d0,-16(a5)
                                ; move.l  d1,-12(a5)
;/*              zx = zx*zx - zy*zy + cx; */
;               zx -= zy2;
^^
                                ; move.l  -32(a5),d2
                                ; move.l  -28(a5),d3
                                ; move.l  -8(a5),d0
                                ; move.l  -4(a5),d1
    REGREG   fsub,zy2,zx        ; jsr     .Psub#
                                ; move.l  d0,-8(a5)
                                ; move.l  d1,-4(a5)
;               zx += cx;
^
                                ; move.l  -40(a5),d2
                                ; move.l  -36(a5),d3
                                ; move.l  -8(a5),d0
                                ; move.l  -4(a5),d1
    REGREG   fadd,cx,zx         ; jsr     .Padd#
                                ; move.l  d0,-8(a5)
                                ; move.l  d1,-4(a5)
;           }
^.13
    add.w    #1,d4
.16
    cmp.w    62(a5),d4
    bcs      iter_loop
```

```
set_result
;               results[nx] = num_iter;
^   move.l   #0,d0
    move.w   d5,d0
    asl.l    #1,d0
    move.l   58(a5),a0
    move.w   d4,(a0,d0.l)
;       }
^.7
    add.w    #1,d5
.10
    cmp.w    66(a5),d5
    bcs    nx_loop
.8
;}
.5
^.17

    SHUTDOWN_881

    movem.l    (sp)+,.4
    unlk    a5
    rts
.2    equ    -48
.4    reg    d4/d5
.3
# 52
|
~ _calc_881 * "{v"
    public    .begin
    dseg
    public  _MC68881_BASE
    end
```

Listing Four
Plot_results.c

```
/*====================< PLOT_RESULTS.C >==================
 *
 *  Copyright (C) 1989
 *  by PiM Publications & Read Predmore
 *  All Rights Reserved
 *  25 March 1989 @ 20:04
 */

#include "mandel.h"

/*====================< PLOT_RESULTS >====================*/
void
plot_results(pwind, results, max_iter, nx_min, nx_max, ny)
struct Window *pwind;
unsigned short  *results, max_iter, nx_min, nx_max, ny;
{
    register  unsigned short i, nx;
    long      pen_color;
    unsigned  short    ncolors;

/*  Subtract 2 since colors 0 and 1 are not used except
    for results[] = max_iter when the color is zero.  */

    ncolors = (1 << pwind->RPort->BitMap->Depth) - 2;

    for(nx=nx_min; nx<nx_max; nx++)
    {
        i = results[nx];
        if(i==max_iter)
            pen_color = 0L;
        else
        {    pen_color = i % ncolors + 2L;
        /*
   printf("In %s, results[%d]=%d, pen_color(%2d,%2d)=%ld\n",
              __FILE__, nx, results[nx], nx, ny,
             pen_color);
         */
        }
        SetAPen(pwind->RPort, pen_color);
        WritePixel(pwind->RPort, (long) nx, (long) ny);
    }
}
/*=====================< FINI >=======================*/
```

Listing Five
Mandel.h

```
/*====================< MANDEL.H >====================
 *
 *  Copyright (C) 1989
 *  by PiM Publications & Read Predmore
 *  All Rights Reserved
 */

#include    <exec/exec.h>
#include    <intuition/intuition.h>
#include    <graphics/gfxbase.h>
#include    <functions.h>
#include    <math.h>
#include    <stdio.h>
#include    <time.h>

/*  Resource Flags  */

#define F_INTUITION    0x000001L
#define F_GRAPHICS     0x000002L
#define F_LAYERSBASE   0x000004L
#define F_MATHTRANS    0x000008L
#define F_CONSOLE      0x000010L
#define F_SCREEN       0x000020L
#define F_WINDOW0      0x000040L
#define F_WINDOW1      0x000080L
#define F_ICON         0x000100L
#define F_GRF_OBJ      0x000200L
#define F_MENUSTRIP    0x000400L
#define F_RASTER       0x000800L
#define F_SUPERBITMAP  0x001000L

#define SCRN_HEIGHT   200
#define SCRN_WIDTH    640
#define SCRN_DEPTH      4
    /*  SuperBitMap parameters.     */
#define SBM_HEIGHT    200L
#define SBM_WIDTH     640L
    /*  IDs for scroll gadgets.     */
#define HORIZ_SCROLL 0x0001
#define VERT_SCROLL  0x0002

struct  BitMap * create_bitmap();
struct  Screen * create_screen();
struct  Window * create_window();

void    free(), *malloc();
void    calc_dble(), calc_881(), close_bitmap(),
        init_color_map(), initialize(), myabort(),
        plot_results(), scroll_SBM(), test_881(),
        update_scroll_gadgets();

#ifndef    MAIN_MODULE
    extern  UBYTE   screen_title[], window_title0[],
                    window_title1[];
    extern  int     Gargc;
    extern  long    cur_resource;
    extern  struct  BitMap  *pSBM;
    extern  struct  Gadget    Horiz_gadget, Vert_gadget;
    extern  struct  GfxBase *GfxBase;
    extern  struct  Image     Horiz_image, Vert_image;
    extern  struct  IntuitionBase *IntuitionBase;
    extern  struct  LayersBase     *LayersBase;
    extern  struct  PropInfo  Horiz_prop,  Vert_prop;
    extern  struct  Screen   *pscreen0;
    extern  struct  Window   *pwindow0;
#endif
```

Listing Six
Mandsubs.c

```
/*====================< MANDSUBS.C >====================
 *
 *  Copyright (C) 1989
 *  by PiM Publications & Read Predmore
 *  All Rights Reserved
 *   26 March 1989 @ 13:19
 */

#include "mandel.h"

/*====================< CLOSE_BITMAP >====================*/
void
close_bitmap(pbitmap)
struct BitMap  *pbitmap;
{
    unsigned char  *planepointer;
    unsigned long   depth,width,height;
    short    i;

    if( pbitmap != 0L)
    {   depth = pbitmap->Depth;
        width = 8 * (pbitmap->BytesPerRow);
        height= pbitmap->Rows;
        for(i=0;i<depth;i++)
        {   planepointer =  pbitmap->Planes[i];
            if(planepointer != 0L)
                FreeRaster(planepointer,width,height);
        }
```

```
        FreeMem(pbitmap, (long) (sizeof (struct BitMap)));
    }
}

/*===================< CREATE_BITMAP >====================*/
struct BitMap *
create_bitmap(width,height,depth)
unsigned long width,height,depth;
{
    short    i, error_flag = FALSE;
    struct    BitMap *pbitmap;
    unsigned char *planepointer;
    unsigned long    bitmapsize;

    pbitmap = 0L;
    if( (depth>0L) && (width>0L) && (height>0L) )
    {   bitmapsize = sizeof (struct BitMap);
        pbitmap = (struct BitMap *) AllocMem(bitmapsize,
                                             MEMF_CHIP);
        if( pbitmap != 0L)
        {   InitBitMap(pbitmap, depth,width,height);
            for(i=0; i<depth; i++)
                pbitmap->Planes[i]=(unsigned char *) 0L;
            for(i=0; i<depth && !error_flag; i++)
            {   planepointer = AllocRaster(width,height);
                if( planepointer )
                    pbitmap->Planes[i] = planepointer;
                else
                    error_flag = TRUE;
            }
        }
    } /* End of if (depth, etc.). */
    if(error_flag == TRUE)
    {   close_bitmap(pbitmap);
        pbitmap = 0L;
    }
    return(pbitmap);
}
/*===================< CREATE_SCREEN >====================*/
struct  Screen *
create_screen(topedge,width,height,depth,  color0,color1,
              mode,name)
short          topedge,width,height,depth;
unsigned char color0,color1;
         char *name;
unsigned short mode;
/*  Mode options are HIRES, INTERLACE, SPRITES and HAM
    which are defined in INCLUDE/GRAPHICS/view.h.  */
{
    struct  NewScreen ns;
    struct  Screen   *pscreen;
    ns.LeftEdge     = 0;
    ns.TopEdge      = topedge;
    ns.Width        = width;
    ns.Height       = height;
    ns.Depth        = depth;
    ns.DetailPen    = color0;
    ns.BlockPen     = color1;
    ns.ViewModes    = mode;
    ns.Type         = CUSTOMSCREEN;
    ns.Font         = NULL;
    ns.DefaultTitle = (UBYTE *) name;
    ns.Gadgets      = NULL;
    ns.CustomBitMap = NULL;
    pscreen = OpenScreen(&ns);
    return(pscreen);
}

/*===================< CREATE_WINDOW >====================*/
struct  Window *
create_window(leftedge,topedge,width,height,  color0,color1,
              flags,IDCMPflags, pFirstGadget, window_title,
              pscreen, psbm)
short          leftedge,topedge,width,height;
unsigned char color0,color1;
unsigned long flags,IDCMPflags;
struct    Gadget *pFirstGadget;
unsigned char  *window_title;
struct  Screen *pscreen;
struct  BitMap *psbm;
{
    struct  NewWindow  nw;
    struct  Window    *pwindow;

    nw.LeftEdge    = leftedge;
    nw.TopEdge     = topedge;
    nw.Width       = width;
    nw.Height      = height;
    nw.DetailPen   = color0;
    nw.BlockPen    = color1;
    nw.Flags       = flags;
    nw.IDCMPFlags  = IDCMPflags;
    nw.Type        = CUSTOMSCREEN;
    nw.FirstGadget = pFirstGadget;
    nw.Title       = window_title;
    nw.CheckMark   = NULL;
    nw.Screen      = pscreen;
```



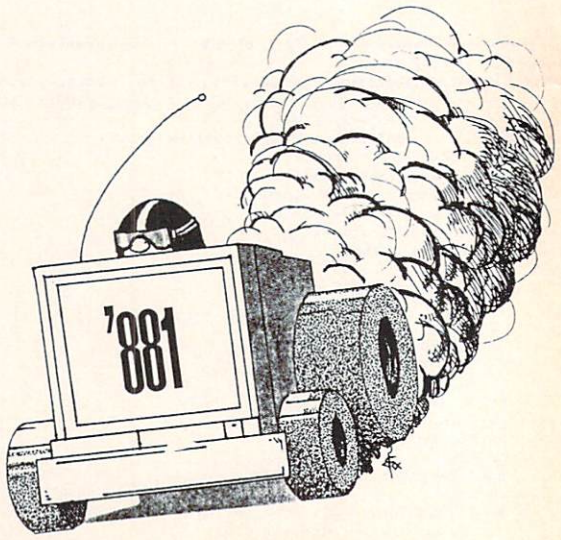
```
    if(flags & SUPER_BITMAP)
        nw.BitMap  = psbm;
    else
        nw.BitMap  = NULL;
    nw.MinWidth    = pscreen->Width/4;
    nw.MinHeight   = pscreen->Height/4;
    nw.MaxWidth    = pscreen->Width;
    nw.MaxHeight   = pscreen->Height;

    pwindow =  OpenWindow(&nw);
    return(pwindow);
}

/*====================< INITIALIZE >====================
 *
 *  Open AMIGA libraries, screen and windows.
 */
void
initialize()
{
    unsigned char  color0, color1;
    unsigned long  flags, IDCMPflags;
    unsigned short viewmode;

    GfxBase = (struct GfxBase *)
              OpenLibrary("graphics.library",0L);
    if(GfxBase == NULL)
        myabort("Can't open graphics.library");
    cur_resource |= F_GRAPHICS;

    IntuitionBase = (struct IntuitionBase *)
                    OpenLibrary("intuition.library",0L);
    if (IntuitionBase == NULL)
        myabort("Can't open intuition.library");
    cur_resource |= F_INTUITION;
```

```
    LayersBase = (struct LayersBase *)
                 OpenLibrary("layers.library",0L);
    if (LayersBase == NULL)
        myabort("Can't open layers.library");
    cur_resource |= F_LAYERSBASE;

    strcpy(screen_title,
    Turbo Mandelbrot (68881) Copyright 1989 by Read Predmore");
    color0 = 0; color1 = 1;
    viewmode = HIRES;
    pscreen0 = create_screen(0, SCRN_WIDTH, SCRN_HEIGHT,
                             SCRN_DEPTH, color0, color1,
                             viewmode, screen_title);
    if(pscreen0 == NULL)
        myabort("Could not open the screen !");
    cur_resource |= F_SCREEN;
    ScreenToBack(pscreen0);
    init_color_map(pscreen0);

    pSBM = create_bitmap(SBM_WIDTH, SBM_HEIGHT,
                         (long) SCRN_DEPTH);
    if(pSBM == NULL)
        myabort("Could not open the Super BitMap !");
    cur_resource |= F_SUPERBITMAP;

    Horiz_prop.Flags = AUTOKNOB | FREEHORIZ;
    Horiz_prop.HorizBody = 0x1FFF;
    Horiz_prop.HorizPot = 0x8000;
    Vert_prop.Flags = AUTOKNOB | FREEVERT;
    Vert_prop.VertBody = 0x1FFF;
    Vert_prop.VertPot = 0x8000;

    flags = WINDOWSIZING | WINDOWDRAG    | WINDOWDEPTH |
            WINDOWCLOSE  | SUPER_BITMAP | GIMMEZEROZERO;

    IDCMPflags = MOUSEMOVE | GADGETDOWN | GADGETUP |
                 CLOSEWINDOW | NEWSIZE
           /*    | SIZEVERIFY */
               ;
    strcpy(window_title0,
" SuperBitMap Window
");
    pwindow0 = create_window(0,11, SCRN_WIDTH,
                             SCRN_HEIGHT-11, color0,color1,
                             flags,IDCMPflags,
                             (struct Gadget *)&Vert_gadget,
                             window_title0, pscreen0, pSBM);
    if(pwindow0 == NULL)
        myabort("Could not open the window !");
    cur_resource |= F_WINDOW0;

    SetRast(pwindow0->RPort, 1L);
    update_scroll_gadgets(pwindow0, &Horiz_prop,&Vert_prop);
    ScreenToFront(pscreen0);
}

/*======================< MYABORT >=========================
 *  Routine which closes windows and screeens and frees up
 *  memory which has been allocated for RasterPorts, etc.
 */
void
myabort(s)
char *s;
{
  if(cur_resource &F_WINDOW0)      CloseWindow(pwindow0);
  if(cur_resource &F_SUPERBITMAP)  close_bitmap(pSBM);
  if(cur_resource &F_SCREEN)       CloseScreen(pscreen0);
  if(cur_resource &F_LAYERSBASE)   CloseLibrary(LayersBase);
  if(cur_resource &F_GRAPHICS)     CloseLibrary(GfxBase);
  if(cur_resource &F_INTUITION)  CloseLibrary(IntuitionBase);
  exit(0);
}

/*====================< SCROLL_SBM >======================*/
void
scroll_SBM(pwind, horiz_prop, vert_prop, dxmax, dymax,
           gadget_id)
struct  Window    *pwind;
struct  PropInfo *horiz_prop,*vert_prop;
long    dxmax, dymax;
USHORT  gadget_id;
{
    long  dx = 0,
          dy = 0;
    if(gadget_id & HORIZ_SCROLL)
    {   dx = (dxmax * ((unsigned long)
              horiz_prop->HorizPot)) / 0xffffL;
        dx = dx - pwind->WLayer->Scroll_X;
    }
    if(gadget_id & VERT_SCROLL)
    {   dy = (dymax * ((unsigned long)
              vert_prop->VertPot)) / 0xffffL;
        dy = dy - pwind->WLayer->Scroll_Y;
    }
    if( (dx != 0L) || (dy !=0L))
        ScrollLayer(pwind->WLayer->LayerInfo,
                    pwind->WLayer, dx, dy);
}
/*================< UPDATE_SCROLL_GADGETS >===============*/
void
update_scroll_gadgets(pwind, horiz_prop, vert_prop)
struct  Window       *pwind;
struct  PropInfo             *horiz_prop,*vert_prop;
{
    horiz_prop->HorizBody = (0xffff * ((unsigned long)
                             pwind->GZZWidth)) /
          (8 * pwind->WLayer->SuperBitMap->BytesPerRow);
    vert_prop->VertBody  = (0xffff * ((unsigned long)
                           pwind->GZZHeight)) /
                           pwind->WLayer->SuperBitMap->Rows;
    RefreshGadgets(pwind->FirstGadget, pwind, 0L);
}
/*======================< FINI >=========================*/
```

Listing Seven
Makefile

```
############################################################
#
#   Makefile to build Mandel_881
#   using Aztec C V3.6a for the AMIGA,
#
#   by Read Predmore,
#   27 March 1989 @ 16:13
#
############################################################
CFLAGS=  +Imand_h -n +FI
OBJS_881 = mand_881.o mandsubs.o plot_results.o \
           calc_dble.o calc_881.o
INCL = mandel.h mand_globals.h

Mandel_881: $(OBJS_881) mand_h
     ln -g -o Mandel_881 $(OBJS_881) -lmt -lc

mand_881.o:  mand_881.c mand_h
     cc  $(CFLAGS)  mand_881.c

mandsubs.o:  mandsubs.c mand_h
     cc  $(CFLAGS)  mandsubs.c

calc_dble.o:  calc_dble.c
   cc  $(CFLAGS)  calc_dble.c

plot_results.o:  plot_results.c mand_h
   cc  $(CFLAGS)  plot_results.c

mand_h: mandel.h
   cc +Hmand_h -a -n -s  mandel.h

calc_881.o:   calc_881.asm MC68881.i
        as  calc_881.asm
```

•AC•



C Notes from the C Group

Avoiding Problems with variable types when passing values between functions

by Stephen Kemp, PLINK ID: SKEMP

During the past year, I have covered the basics of programming in C . I have tried to emphasize the importance of using the proper "type" variables when passing parameters between functions. This month, I want to make that point again, and discuss some ways you can avoid problems.

Assume you have a function that expects to receive a long integer as a parameter. If you accidentally call the function with a short integer, your program will not work properly. The terrible thing about a subtle error like this is that the failure may not occur immediately. And debugging an error like this can be a nightmare. Simple type mismatches have caused more than just erroneous program output . Sometimes they can cause far worse problems, like a "locked up" machine or even a trashed hard disk.

Now I'm not saying this to frighten you away from C. Remember, every programming language has its share of potential hazards and pitfalls. As a programmer, it will be your responsibility to ensure the program's integrity. And depending on your choice of C compilers, there are a couple of "tools" available to help you.

The most obvious thing you can do to avoid problems like these is maintain a stack of current program listings. When uncertainties arise about the parameters a fdunction requires, just pull out the proper listing. Unfortunately, listings have some disadvantages. They are not always convenient to keep updated. You must make a place to keep them orderly; and most programs rely heavily upon library functions for which you will not have the source.

The next alternative—one I use on most occasions—is to maintain a simple list containing the function's parameters and "return" types. I like to keep the list sorted alphabetically, or divided into logical groups (i.e., string functions, memory functions, disk functions, etc.) Printed out and posted on a wall nearby, a single list is much handier than a complete set of listings.

Whether you keep listings or a function list, there will be times when a function must be called with a parameter that does not quite match the type the function requires. If you find yourself in one of these situations, there are a few things you can do. One is to alter the function, perhaps by adding another variable of the proper type, and calling the function with the proper parameter type. A second, and usually better, option is to use "casting".

Casting

Casting is a means of overriding a variable's declared type by indicating the desired variable type in parenthesis prior to the variable's label in a statement. All C compilers should support casting (I have not found one that doesn't). The example involving the use of a short variable in a long parameter's place can be repaired with a simple cast operation. Here is a little code to demonstrate.

```
function_a()
{
    short var;

.....

    function_b( (long) var );
    /* call to function with long */

.....

}
```

Although this may seem like an odd example, it is not that uncommon. The function (function_a) has a variable declared as a short. Assume function_b expects a parameter of type "long". Since shorts and longs are different sizes, the cast is needed so the correct number of bytes will be passed to the function. The compiler actually promotes the variable to the desired type prior

(continued)

to pushing the parameter on the stack. The cast only operates on the next logical item occurring in the statement after it is declared. (Note: When dealing with "signed" values, the compiler will promote the sign also.)

Casting works just as well with non-integer types. It can be especially handy when using pointers. Many compilers will recognize (and issue warnings) when pointer types are "mixed". Although some warning messages emitted from compilers are generated in harmless situations, I believe a good programmer will make every effort to make programs that are error and warning-free.

The following example shows two types of pointers, and demonstrates how to cast them in statements. Assume the function is designed to examine the individual bytes (chars) of a long variable which is pointed at by the other pointer.

```
.....

char *cptr;       /* character pointer */
long *lptr;       /* long pointer */

....

cptr = (char *) lptr;
    /* set the char pointer */

....
    /* code to examine and perhaps change the pointer*/

lptr = (long *) cptr;         /* set the long pointer */
```

The trouble with casting is that you must remember to do it. If you forget, you might not get a message from the compiler, but the program could fail miserably. Fortunately, many compilers now support a feature to aid in the fight for an error-free program. This feature is called prototyping., and it is similar to casting, but goes a large step further. When you provide the compiler with a prototype for a function, the compiler examines each reference to the prototyped item to ensure proper use. This can be an invaluable feature, and I highly recommend you ask about it when purchasing a C compiler. A function prototype looks similar to a function declaration, but merely serves as a definition. Consider this example:

```
char * function_z( long x, long *y );
    /* prototype for function*/
```

This definition tells the compiler exactly what type of parameters function_z expects, and what type of value it will return. Looking back at the list of functions I recommended, you can see that it is actually a list of function prototypes. Tailored to meet the requirements of a given compiler that supports prototyping, a prototype "file" will catch/prevent many potential programming errors. Referring back to the prototype of function_z, if the compiler ran across the following code, for example, it would emit a number of warnings/errors.

```
function_a()
{
    short var;
    char string[sizeof(long)];
    /* a char array to hold a long */
    long *lptr;

    ....
    /* assume var is assigned some value */

    lptr = function_z( var, string);
    /* call the function */

    ....
}
```

Again, this is not something you should do on a regular basis, but the unorthodox is occasionally necessary. Corrected with casting, the statement looks like this:

```
lptr = (long *) function_z( (long) var, (long *) string);
```

Prototyping can prevent many disasters. If your C compiler does not support it, don't despair just yet. A separate program, usually called "lint", can examine your program's code with the same strong type-checking provided by prototyping. I am not sure why it's called lint (unless it is analogous to the lint trap in a washer or dryer), but a good lint program can be a lifesaver. If your package does not support prototyping, check into a good lint program. Most compilers don't come with a lint program, so you will have to purchase one separately. You should be able to find a lint program in a good software store or mail-order ad, but to avoid any compatibility problems, be sure to get one which explicitly states that it supports your C compiler .

As I have stated, lint has the same capabilities as prototyping. Unlike prototyping, lint must be run separately over your program modules. It will emit the same type errors and warnings as a prototyping compiler . If you redirect the output to the printer or a disk file, you can then use the output as a reference when fixing the problem areas.

Programming without problems is a tough task. Fortunately, we have not been thrown to the lions without weapons. Incorporating the suggestions outlined here will not guarantee you bug-free programs, but the chances for success are certainly improved. If you are unsure whether your compiler supports some of the operations I have discussed, dust off the manual and find out. If it does, start experimenting.

•AC•

Amiga Video: 90%Preparation

Prepare your video for summertime fun

by Otto Focus

[As a special feature this month, AC is providing the insights of a rather obscure and unconventional video hobbyist. Due to his unorthodox manner and style, he has permitted the use of this article with a pseudonym only. We can only offer the following information on this character: He practices what he preaches-sometimes.

Even though the author is a slight off the mark, his comments on video vacationing and video preparedness are timely. The Summer is an excellent time to do more with your video equipment. Summer vacations create the perfect opportunity to travel and record new sights and sounds.

We Amiga users want to use our Amigas to produce well edited videos which we will proudly show our friends and relatives. (It's only fair, we've seen their slides.) In light of this opportunity, AC will do a special video supplement in our August issue. We offer this article as a means to prepare your tapes before the editing process. ED.]

BE PREPARED!

When I was a boy scout, the one thing the scout leaders drilled into our heads was "Be Prepared." This is a lesson I have brought into my adult life with extreme enthusiasm. I am never able to pack a single bag for a trip even if just for two days. I am constantly "stocking up" on everything from shampoo to batteries. If I am driving any distance, I store tools, radios, blankets, water, medicines, books, tapes, and just about any other thing that will fit into my car until it looks as if I am packing for the Santa Fe Trail. In short, I don't want to be caught short.

You may wonder what this has to do with Video. Everything! I sincerely believe that programming, photography, and video all share the same basic three principles: know your tools, be prepared, and "garbage in garbage out."

Know Your Tools

Your first line of preparation is to know what equipment you have, what your equipment is capable of doing, and how to use each piece. It is never a good idea to begin a serious project without at least being familiar with your equipment. Before a serious recording effort, conduct a practice recording session around your home and neighborhood. (If your neighbors begin giving you funny looks, tell them you're looking for UFO's. They will probably think you're a little strange, but they won't know about your computer habits.)

Since you are reading a video article in Amazing Computing, it is safe to assume you will eventually want to use the Amiga to edit and postproduce your video. Practicing with this test tape will give you a sense of accomplishment as well as competence with your equipment. These tests will also prepare your judgement in what to shoot and how to shoot it.

You should attempt to attain a strong sense of how each piece of Amiga equipment you plan to use will fit into your final production. Remember, we are not trying to take the spontaneity out of your work, but to arm your creative side with all the available information you may require. This will give you the tools you will need to make those quantum leaps of imagination and create a video worthy of your time and effort.

Since each video camera tends to be unique in its features, it is best to consult your manual and any other material you have been able to collect on your equipment. Be aware of the lighting requirements, tape sizes, battery life, handling requirements and other necessities of your equipment. Are special adaptors required for use with alternating current, how long does a battery take to recharge, what actually makes the auto focus work and does it? These are the questions that once answered, give you a strong competent feeling that you will get the results you want.

A Video of What?

Once you know how your video equipment works, you should decide what to shoot. The purpose of this article in a Summer issue of Amazing Computing is to catch all the Amiga videophiles, with camcorders in hand as they prepare for their vacations. To obtain the desired result, you must first decide what you want. Even a vacation video can be much more entertaining if you prepare your thoughts in advance.

By anticipating what you will be recording, you will have a better understanding of which piece of equipment you should bring. This will make your "on location shooting" easier if you are not concerned with an enormous amount of clutter.

Always anticipate the problems a certain area may give you. Will you need long periods of time on battery power? Will you be in low light situations? Will the environment be a hazard to your equipment? Will your current case be helpful and protective, or just hard to manage.

Let's assume you're headed for the woods, an amusement park, a classical performance outside, a museum, the beach, Aunt Milley's, the drag races, dog show, steam train rides, or wherever. Most of the fun with the new camcorders is that you are not tied to an electrical outlet and, unlike the old 3 minute movie films, you can obtain over two hours of recording time on one tape, provided you have enough battery life.

Battery life is one of the easiest problems to solve. Unfortunately, all you need is money. Most of the higher priced cameras come with replaceable batteries. These units can be purchased for around $50.00. However, they can provide more than 2 hours of recording time while taking approximately the same amount of time to recharge (individual products will vary, please check your equipment's requirements). When you add up the cost of your current Video equipment and the cost of your vacation, you may decide not to go on your trip, or you may decide that a few dollars more for that extra battery life will mean hours more of freedom and entertainment.

Most cameras make the batteries extremely easy to change. This means you will not lose a lot of time and can continue the fun immediately. Besides, there is nothing like the thrill of pulling a tired battery from your camera and snapping a new charge into its place. It is a combination of being an evening news reporter and Rambo.

Obviously, if you expect to record in a low light setting, make sure your camera can produce clear shots. The newer full feature cameras have excellent low light ability, however some genlocks and frame grabbers have difficulty producing a good duplicate of the original.

This is where knowing your equipment truly comes into play. By testing your equipment beforehand, you will have a better sense of what you will be able to do and what will fail. And you'll save severe frustrations later.

Weathering the elements

The next two questions are closely related, environmental factors and your case. Your case should protect your camera and assorted equipment from the hazards of the location, sun, water, sand, heat, etc.

Your case should be big enough to hold your required accessories (alone or augmented by an equally protective sister case). The case should also be capable of traveling where you need to go. If you are traveling on a commercial airline, be aware of the restrictions of carry on luggage. Most airlines in the US allow you to have two carry on bags. Each bag must fit under the seat in front of you, or in an overhead compartment.

I suggest you store the case in the space under the seat in front of you. The idea that sometime during the flight, the overhead compartment will open and your camera will drop on a fellow passenger's head is too much. I have found that a good duffle bag fits in this area and makes an excellent case.

Your duffle bag/camera case can be "lined" inside with extra video tapes, batteries, etc. and still leave room for a tape player, tapes, a book and your favorite computer magazine, Amazing Computing. (I warned you I tend to overpack.) Everything will fit neatly in the bag, but beware, it may become extremely heavy.

Your last consideration for equipment should be all the things you need to do your job better. A lens cleaner and air brush to keep your camera clean. Good tapes will not only make you pictures better, but it will also preserve the heads in your camera and the cost is generally just a few dollars more.

A tripod is a definite consideration. Depending on where you are shooting a tripod can be a help or a bother. While it is difficult to use a tripod on a roller coaster, it is equally unnerving to watch an unsteady camera image caused by the shaking hands of a weekend camera operator. If you intend to add computer graphics to the video, the resulting image can create far more severe effects than a roller coaster ride.

A tripod will anchor your camera for those long smooth shots used in your credits and other special effects. It also makes it possible to switch from camera operator to fellow performer and allow you to have fun with the rest of the family.

Think of The Final Effect

Always remain in touch with your final product. Keep in mind how you will eventually use the shots you are taking. Be aware of the different possibilities you have and judge which shots would make these ideas and your final product more interesting.

Take your time. With most recorders, each tape provides between one to two hours of recording time. If you take enough footage, you will have a wealth of material to choose during the editing process. Try not to record quick, short shots. Use the recorder in longer periods and then edit scenes together for a more fluid look. This is especially true if you want to add a title or credits to your video.

Record a long section of that perfect beach scene, sunset, or mountain view to make a backdrop for your Amiga lettering. Remember to take more recordings than you need—a good editor wants far more material than he MUST use. This gives you a choice in the editing process.

Summary

My comments are not meant to stifle your creative edge. I want you to have the opportunity to produce your best and you will only accomplish this if you know what to do and how to do it. Remember, your family will dutifully sit through your vacation video the first time, they want to see themselves and they will feel obligated to you. However, when they ask to see the tape again, they want a copy for their friends, or they ask you to record a special holiday or family event, you will know your work is well appreciated.

•AC•

*(**UPS**, continued from page 86)*

second sample tried to cold boot the expanded Amiga computer system, but it was incapable of doing so. That it even attempted to do so, must have been part of the defect related to the sensing of its AC / battery condition.

This second sample otherwise functioned properly under both AC and battery backup conditions in the subsequent in-use tests. When the DRS unit goes to battery backup, if the battery is fully charged, all ten battery meter LEDs light up. This quickly drops off by one or two LEDs. The remaining green LEDs extinguish slowly, one at a time, as battery backup operation continues. I obtained a third UPS sample , and tested it with in-use timed tests. In operation, all three samples of the DRS-350 made fairly loud audible noises from their power transformers. This vibration could easily be felt by touching the outer case. This appeared to be normal for this design.

The company stated that the problems in the first two samples were the result of defective cable assemblies. There were several units with less than perfect plug-in cables when our samples were sent out.—reportedly some 12 - 15 unit s before the cause of the problem was identified. The third sample of the DRS-350 tested had no defects at all.

DRS Power Products warns that a load must be plugged into the unit when it is turned on. To have the same test conditions, across all brands of UPS units, the units were tested without a load—to simulate all external equipment being off — during preliminary turn-on and charging conditions. Only one other brand, Kalglo, carried a warning not to operate it without a load.

The DRS-350 gets warm on the top middle of the case during battery backup operation. It was fairly cool during battery recharging, even after full battery depletion. The DRS circuit uses a special high rate charge circuit so battery charging takes only 6 - 8 hours.

Unique to the DRS units, are the LED " LOAD CENTER " displays on the front panel. These 10 level bar graphs can provide extremely useful information on the actual AC load and battery condition. It's one of the best features on any UPS. Units from other companies lack such an accurate method of estimating both AC load level and battery condition.

The rear panel of one DRS-350 also had a DB-25 connector. This connector was part of the " Network Option ", consisting of two internal relays, and the DB-25 connector. This provides two separate signals, which can indicate both AC power failure, and imminent UPS shutdown to the file server on a computer network.

Originally, the DRS-350 didn't come with a manual, just a specification sheet containing some technical information, a one-page Quick Start Guide, a warranty card, and some information on what to do if the unit is physically damaged. It was not comprehensive. We recieved a 27-page User Manual befor the tests were completed. The manual is informative and comprehensive, but repititious. The same information often appears two or three times in different sections. Most people would find its information to be quite adequate. There is a two year warranty period for the original owner.

The second sample of the DRS-350 UPS backed up the Amiga A-1000 / Thomson / equivalent expanded computer system for 36 minutes, 40 seconds; the third sample lasted 37 minutes, 48 seconds. Overall, the unit is very impressive, if you get one that is fully functional. List Price: $499.00 Unit is available discounted.

Summary

THE DRS - 350 UPS

Positive attributes: AC load and battery condition bar graph displays, medium size, optional network option, 2 year warranty.

Negative attributes: Very deep aluminum case, no external battery jacks, no alarm turn-off switch, uses plastic encapsulated output devices, significant transformer hum on AC and more on battery backup, does not cold boot in absence of AC power, defects present in 2 of 3 samples tested.

[Look for more UPS reviews by Stephen L Bender in upcoming issues of AC—Ed.]

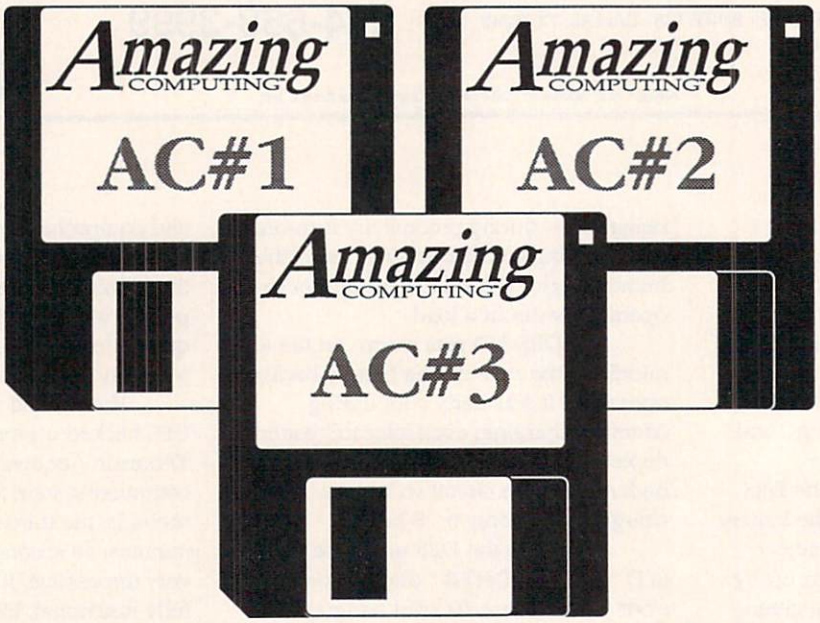
•AC•

Sometimes three isn't a crowd.

Amazing on Disk

Source Listings and Executables
from the pages of Amazing Computing!

***Only $6.00 per disk** ($7.00 for Non-Subscribers)*



Amazing on Disk #1, #2 & #3

Amazing on Disk #1 complete source listings from AC 3.8 and 3.9

Tumbling Tots: A game written in assembler (AC V3.8)
Modula-2: FFP & IEEE math routines (AC V3.8).
Gels in Multi-Forth: Part I & II (AC V3.9).
The Complete CAI Authoring System in AmigaBASIC (AC V3.9).

and a few freely redistributable goodies like:

B-Spread
MenuEd & VGad: programming tools featured in "The Developing Amiga" (AC V3.8).

Amazing on Disk #2 complete source listings from AC 4.4

GetArgs.MSB: Passing arguments to AmigaBASIC from CLI.
Shared Library Assembler Header: Use shared libraries, written in 'C'.
MultiSort: Sorting and intertask communication in Modula-2.
'881 Math Part I: Part one on programming the 68881 Math Coprocessor Chip.

Amazing on Disk #3 complete source listings from AC 4.5 and 4.6

Digitized Sound: A Modula 2 program that plays sound files (AC V4.5).
'881 Math Part II: Part two on programming the 68881. Shows how with fractal drawing program (AC V4.5).
Insta Sound: Tapping the Amiga's sound from AmigaBASIC (AC V 4.5).
MIDI Out: A MIDI drill program written in C that you can add features to *(AC V4.5).*
At Your Request: A program that shows you how to access system requestors from AmigaBASIC *(AC V 4.6).*
Diskless Compiler: See how you can set up a compiler enviroment that doesn't need floppies *(AC V4.6).*

(Order Form on inside rear cover)

The AMICUS & Fred Fish Public Domain Software Library

This software is collected from user groups and electronic bulletin boards around the nation. Each Amicus disk is nearly full, and is fully accessible from the Workbench. If source code is provided for any program, then the executable version is also present. This means that you don't need the C compiler to run these programs. An exception is granted for those programs only of use to people who own a C compiler.

The Fred Fish disk are collected by Mr. Fred Fish, a good and active friend of the Amiga.

Note: Each description line below may include something like 'S-O-E-D', which stands for 'source, object file, executable and documentation'. Any combination of these letters indicates what forms of the program are present. Basic programs are presented entirely in source code format.

AMICUS Disk 1

ABasic programs: Graphics

3DSolids	3d solids modeling prog. w/sample data files
Blocks	draws blocks
Cubes	draws cubes
Durer	draws pictures in the style of Durer
FScape	draws fractal landscapes
Hidden	3D drawing program, w/ hidden line removal
JPad	simple paint program
Optical	draw several optical illusions
PaintBox	simple paint program
Shuttle	draws the Shuttle in 3d wireframe
SpaceArt	graphics demo
Speaker	speech utility
Sphere	draws spheres
Spiral	draws color spirals
ThreeDee	3d function plots
Topography	artificial topography
Wheels	draws circle graphics
Xenos	draws fractal planet landscapes

ABasic programs: Tools

AddressBook	simple database program for addresses
CardFile	simple card file database program
Demo	multiwindow demo
KeyCodes	shows keycodes for a key you press
Menu	run many ABasic programs from a menu
MoreColors	way to get more colors on the screen at once, using aliasing
shapes	simple color shape designer Speakit speech and narrator demo

ABasic programs: Games

BrickOut	classic computer brick wall game
Othello	also known as 'go'
Saucer	simple shoot-em-up game
Spelling	simple talking spelling game
ToyBox	selectable graphics demo

ABasic programs: Sounds

Entertainer	plays that tune
HAL9000	pretends it's a real computer
Police	simple police siren sound
SugarPlum	plays "The Dance of the Sugarplum Fairies"

C programs:

ATerm	simple terminal program, S-E
cc	aid to compiling with Lattice C
decvnt	opposite of CONVERT for cross developers
Dotty	source code to the 'dotty' window demo
echox	unix-style filename expansion, partial S,O-D
fasterfp	explains use of fast-floating point math
FixDate	fixes future dates on all files on a disk, S-E
treedraw	simple Workbench drawing prog.,S-E
GfxMem	graphic memory usage indicator, S-E
Grep	searches for a given string in a file with
docs.	ham shows off the hold-and-modify method of color generation
IBM2Amiga	fast parallel cable transfers between an IBM and an Amiga
Mandel	Mandelbrot set program, S-E
moire	patterned graphic demo, S-E
objfix	makes Lattice C object file symbols visible to Wack, S-E
quick	quick sort strings routine
raw	example sample window I/O
setiace	turns on interlace mode, S-E
sparks	qix-type graphic demo, S-E

Other executable programs:

SpeechToy	speech demonstration
WhichFont	displays all available fonts

Texts:

68020	describes 68020 speedup board from CSA
Aliases	explains uses of the ASSIGN command
Bugs	known bug list in Lattice C 3.02
CLICard	reference card for AmigaDOS CLI
CLICommands	guide to using the CLI
Commands	shorter guide to AmigaDOS CLI commands
EdCommands	guide to the ED editor
Filenames	AmigaDOS filename wildcard conventions
HalfBright	explains rare graphics chips that can do more colors
ModemPins	description of the serial port pinout
RAMdisks	tips on setting up your RAM: disk
ROMWack	tips on using ROMWack
Sounds	explanation of Instrument demo sound file format
Speed	refutation of Amiga's CPU and custom chip speed
WackCmds	tips on using Wack

AMICUS Disk 2

C programs:

alib	AmigaDOS object library manager , S-E
ar	text file archive program, S-E
fixobj	auto-chops executable files
shell	simple CLI shell, S-E
sq, usq	file compression programs, S-E
YachtC	a familiar game, S-E
Make	a simple 'make' programming utility, S-E
Emacs	an early version of the Amiga text editor, S-E-D

Assembler programs:

bsearch.asm	binary search code
qsort.asm	Unix compatible qsort() function, source and C test program
setjmp.asm	setjmp() code for Lattice 3.02
SVprintf	Unix system V compatible printf()
trees.o	Unix compatible tree() function, O-D

(This disk formerly had IFF specification files and examples. Since this spec is constantly updated, the IFF spec files have been moved to their own disk in the AMICUS collection.)

John Draper Amiga Tutorials:

Animate	describes animation algorithms
Gadgets	tutorial on gadgets
Menus	learn about Intuition menus

AMICUS Disk 3

C programs:

Xref	a C cross-reference gen., S-E
6bitcolor	extra-half-bright chip gfx demo, S-E
Chop	truncate (chop) files down to size, S-E
Cleanup	removes strange characters from text files
CR2LF	converts carriage returns to line feeds in Amiga files, S-E
Error	adds compile errors to a C file, S
Hello	window ex. from the RKM, S
Kermit	generic Kermit implementation, flakey, no terminal mode, S-E
Scales	sound demo plays scales, S-E
SkewB	Rubik cube demo in hi-res colors, S-E

AmigaBasicProgs(dir)

Automata	cellular automata simulation
CrazyEights	card game
Graph	function graphing programs
WitchingHour	a game

ABasiC programs:

Casino	games of poker, blackjack, dice, and craps
Gomoku	also known as 'othello'
Sabotage	sort of an adventure game

Executable programs:

Disassem	a 68000 disassembler, E-D
DpSlide	shows a given set of IFF pictures, E-D
Arrange	a text formatting program, E-D

Assembler programs:

Argoterm	terminal program with speech and Xmodem, S-E

AMICUS Disk 4 Files from the original Amiga Technical BBS

Note that some of these files are old, and refer to older versions of the operating system. These files came from the Sun system that served as Amiga technical support HQ for most of 1985. These files do not carry a warranty, and are for educational purposes only. Of course, that's not to say they don't work.

Complete and nearly up-to-date C source to 'image.ed', an early version of the Icon Editor. This is a little flaky, but compiles and runs.

An Intuition demo, in full C source, including files: demomenu.c, demomenu2.c, demoreq.c, getascii.c, idemo.c, idemo.guide, idemo.make, idemoall.h, nodos.c, and txwrite.c

addmem.c	add external memory to the system
bobtest.c	example of BOB use
consoleIO.c	console IO example
creaport.c	create and delete ports
creastdi.c	create standard I/O requests
creatask.c	creating task examples
diskio.c	example of track read and write
dotty.c	source to the 'dotty window' demo
dualplay.c	dual playfield example
flood.c	flood fill example
freemap.c	old version of 'freemap'
getltools.c	tools for VSprites and BOBs
gfxmem.c	graphic memory usage indicator
hello.c	window example from RKM
inputdev.c	adding an input handler to the input stream
joystik.c	reading the joystick
keybd.c	direct keyboard reading
layertes.c	layers examples
mousport.c	test mouse port
ownlib.c, ownlib.asm	example of making your own library with Lattice
paratest.c	tests parallel port commands
seritest.c	tests serial port commands
serisamp.c	example of serial port use
prinintr.c	sample printer interface code
prtbase.h	printer device definitions
regintes.c	region test program
setiace.c	source to interlace on/off program
setparallel.c	set the attributes of the parallel port
SetSerial.c	set the attributes (parity, data bits, etc.) of the serial port
singplay.c	single playfield example
speechtoy.c	source to narrator and phonetics demo
timedely.c	simple timer demo
timer.c	exec support timer functions
timrstuf.c	more exec support timer functions
WhichFont.c	loads and displays all available system fonts

process.i and prtbase.i assmebler include files:

autorqstr.txt	warnings of deadlocks with autorequesters
consoleIO.txt	copy of the RKM console I/O chapter
diskfont.txt	warning of disk font loading bug
fullfunc.txt	list of #defines, macros, functions
inputdev.txt	preliminary copy of the input device chapter

License information on Workbench distribution license

printer pre-release copy of the chapter on printer drivers, from RKM 1.1 v11fd.txt 'diff' of .fd file changes from version 1.0 to 1.1

v28v1.diff 'diff' of include file changes from version 28 to 1.0

AMICUS Disk 5 Files from the Amiga Link / Amiga Information Network

Note that some of these files are old, and refer to older versions of the operating system. These files are from Amiga Link. For a time, Commodore supported Amiga Link, aka AIN, for online developer technical support. It was only up and running for several weeks. These files do not carry a warranty, and are for educational purposes only. Of course, that's not to say they don't work.

A demo of Intuition menus called 'menudemo', in C source

whereis.c	find a file searching all subdirectories
bobtest.c	BOB programming example
sweep.c	sound synthesis example

Assembler files:

mydev.asm	sample device driver
mylib.asm	sample library example
mylib.i	
mydev.i	
asmsupp.i	
macros.i	assembler include files

Texts:

amigatricks	tips on CLI commands
extdisk	external disk specification
gameport	game port spec
parallel	parallel port spec
serial	serial port spec
v1.1update	list of new features in version 1.1
v1.1h.txt	'diff' of include file changes from version

Files for building your own printer drivers, including dospecial.c, epsondata.c, init.asm, printer.c, printer.link, printertag.asm, render.c, and wait.asm. This disk does contain a number of files describing the IFF specification. These are not the latest and greatest files, but remain here for historical purposes. They include text files and C source examples. The latest IFF spec is elsewhere in this library.

AMICUS Disk 6 IFF Pictures

This disk includes the DPSlide program, which can view a given series of IFF pictures, and the 'showpic' program, which can view each file at the click of an icon. The pictures include a screen from ArticFox, a Degas dancer, the guys at Electronic Arts, a gorilla, horses, King Tut, a lighthouse, a screen from Marble Madness, the Bugs Bunny Martian, a still from an old movie, the Dire Straits moving company, a screen from Pinball Contruction Set, a TV newcaster, the PaintCan, a world map, a Porsche, a shuttle mission patch, a tyrannosaurus rex, a planet view, a VISA card, and a ten-speed.

AMICUS Disk 7 DigiView HAM demo picture disk

This disk has pictures from the DigiView hold-and-modify video digitizer. It includes the ladies with pencils and lollypops, the young girl, the bulldozer, the horse and buggy, the Byte cover, the dictionary page, the robot and Robert. This includes a program to view each picture separately, and all together as separate, slidable screens. The 'seeibm' program, to turn any screen into an IFF picture.

AMICUS Disk 8

C programs:

Browse	view text files on a disk, using menus S-E-D
Crunch	removes comments and white space from C files, S-E
IconExec	EXECUTE a series of commands from Workbench S-E
PDScreen Dump	dumps Rastport of highest screen to printer
SetAlternate	sets a second image for an icon, when clicked once S-E
SetWindow	makes windows for a CLI program to run under Workbench S-E
SmallClock	a small digital clockin a window menu bar
Scrimper	the screen printer in the fourth AC S-E

Amiga Basic Programs:

(Note: Many of these programs are present on AMICUS Disk 1. Several of these were converted to Amiga Basic, and are included here.)

AddressBook	a simple address book database
Ball	draws a ball
Cload	program to convert Compuserve hex files to binary, S-D
Clue	the game, Intuition driven
ColorArt	art drawing program
DeluxeDraw	the drawing program in the 3rd AC, S-D
Eliza	conversational computer psychologist
Othello	the game, as known as 'go'
RatMaze	3D ratmaze game
ROR	boggling graphics demo
Shuttle	draws 3D pictures of the space shuttle
Spelling	simple spelling program
YoYo	wierd zero-gravity yo-yo demo, tracks yo-yo to the mouse

Executable programs:

3Dcube	Modula-2 demo of a rotating cube
AltIcon	sets a second icon image, displayed when the icon is clicked
AmigaSpell	a slow but simple spell checker, E-D
arc	the ARC file compression progra must-have for telecom, E-D
Bertrand	graphics demo
disksalvage	prog. to rescue trashed disks, E-D
KwikCopy	a quick but nasty disk copy program: ignores errors, E-D
LibDir	lists hunks in an object file E-D
SaveILBM	saves any screen as IFF pic.E-D ??
ScreenDump	shareware screen dump prog, E only
StarTerm	version 2.0, term program, XmodemE-D

Texts:

LaticeMain	tips on fixing _main.c in Lattice
GDiskDrive	make your own 5 1/4 drive
GuruMed	explains the Guru numbers
Lat3.03bugs	bug list of Lattice C version 3.03
MForgeRev	user's view of the MicroForge HD
PrintSpooler	EXECUTE-based print spool prog.

.BMAP files:

These are the necessary links between Amiga Basic and the system libraries. To take advantage of the Amiga's capabilities in Basic, you need these files. BMAPs are included for 'clist', 'console', 'diskfont', 'exec', 'icon', 'intuition', 'layers', 'mathffp', mathieeedoubas', 'mathieeesingbas', 'mathtrans', 'potgo', 'timer' and 'translator'.

AMICUS Disk 9

Amiga Basic Programs:

FlightSim	simple flight simulator program
HuePalette	explains Hue, Saturation, & Intensity
Requester	ex. of requesters from Amiga Basic
ScrollDemo	demonstrates scrolling capabilities
Synthesizer	sound program
WorldMap	draws a map of the world

Executable programs:

Boing!	latest Boing! demo,with selectable speed,E
Brush2C	converts an IFF brush to C data instructions, initialization code, E
Brush2Icon	converts IFF brush to an icon, E
Dazzle	graphics demo, tracks to mouse, E
DeciGEL	assembler program for stopping 68010 errors, S-E-D
Klock	menu-bar clock and date display, E
life	the game of life, E
TimeSet	Intuition-based way to set the time & date
EMEmacs	another Emacs, more oriented to word processing, S-E-D
MyCLI	a CLI shell, works without the Workbench, S-E-D

Texts:

FnctnKeys	read function keys from Amiga Basic
HackerSIn	explains how to win the game 'hacker'
Ist68010	guide to installing a 68010 in your Amiga
Boing!	latest Boing! demo,with selectablespeed, E
Brush2C	converts an IFF brush to C data instructions, initialization code, E
Brush2Icon	converts IFF brush to an icon, E
Dazzle	graphics demo, tracks to mouse, E
DeciGEL	assembler program for stopping 68010 errors, S-E-D
Klock	menu-bar clock and date display, E
life	the game of life, E
TimeSet	Intuition-based way to set the time date,
EMEmacs	another Emacs, more oriented to word processing, S-E-D
MyCLI	a CLI shell, works without the Workbench, S-E-D

For PDS orders, please use form on page CIII

Texts:

FnctnKeys	explains how to read function keys from Amiga Basic
HackerSin	explains how to win the game 'hacker'
Ist68010	guide to installing a 68010 in your Amiga
PrinterTip	sending escape sequences to your printer
StartupTip	tips on setting up your startup-sequence file
XfrmrReview	list of Transformer programs that work

Printer Drivers:
Printer drivers for the Canon PJ-1080A, the C Itoh Prowriter, an improved Epson driver that eliminates streaking, the Epson LQ-800, the Gemini Star-10, the NEC 8025A, the Okidata ML-92, the Panasonic KX-P10xx family, and the Smith-Corona D300, with a document describing the installation process.

AMICUS Disk 10 Instrument sound demos
This is an icon-driven demo, circulated to many dealers. It includes the sounds of an acoustic guitar, an alarm, a banjo, a bass guitar, a boink, a calliope, a car horn, claves, water drip, electric guitar, a flute, a harp arpegio, a kickdrum, a marimba, a organ minor chord, people talking, pigs, a pipe organ, a Rhodes piano, a saxophone, a sitar, a snare drum, a steel drum, bells, a vibrophone, a violin, a wailing guitar, a horse whinny, and a whistle.

AMICUS Disk 11

C programs

dirutil	Intuition-based, CLI replacement manager S-E
cpri	shows and adjusts priority of CLI processes, S-E
ps	shows info on CLI processes, S-E
vidtex	displays Compuserve RLE pics, S-E

AmigaBasic programs

pointered	pointer and sprite editor program
optimize	optimization ex ample from AC article
calendar	large, animated calendar, diary and date book program
amortize	loan amortizations
brushtoBOB	converts small IFF brushes to AmigaBasic BOB OBJECTS
grids	draw and play waveforms
hilbert	draws Hilbert curves
madlib	mad lib story generator
mailtalk	talking mailing list program
meadows3D	3D graphics program, from A C™ article
mousetrack	mouse tracking example in hires mode
slot	slot machine game
tictactoe	the game
switch	pachinko-like game
weird	makes strange sounds

Executable programs

cp	unix-like copy command, E
cls	screen clear, S-E
diff	unix-like stream editor uses 'diff' output to fix files
pm	chart recorder performances indicator

Assembler programs

cls	screen clear and CLI arguments example

Modula-2

trails	moving-worm graphics demo
caseconvert	converts Modula-2 keywords to uppercase
Forth	Breshehan circle algorithm example
Analyze	12 templates for the spreadsheet Analyze

There are four programs here that read Commodore 64 picture files. They can translate Koala Pad, Doodle, Print Shop and News Room graphics to IFF format. Getting the files from your C-64 to your Amiga is the hard part.

AMICUS Disk 12

Executable programs

blink	'alink' compatible linker, but faster, E-D
clean	spins the disk for disk cleaners, E-D
epsonset	sends Epson setttings to PAR from menu E-D
showbig	view hi-res pics in low-res superbitmap, E-D
speaktime	tell the time, E-D
undelete	undeletes a file, E-D
cnvapidhm	converts Apple][low, medium and high res pictures to IFF, E-D
menued	menu editor produces C code for menus, E-D
quick	quick disk-to-disk nibble copier, E-D
quickEA	copies Electronic Arts disks, removes protection, E-D
txed 1.3	demo of text editor from Micrositths,E-D

C programs

spin3	rotating blocks graphics demo, S-E-D
popcli	start a new CLI at the press of a button, like Sidekick, S-E-D
vsprite	VSprite example code from Commodore, S-E-D
AmigaBBS	Amiga Basic bulletin board prog., S-D

Assembler programs

star10	makes star fields like Star Trek intro,S-E-D

Pictures

Mount Mandelbrot	3D view of Mandelbrot set
Star Destroyer	hi-res Star Wars starship
Robot	robot arm grabbing a cylinder

Texts

vendors	Amiga vendors, names, addresses
cardco	fixes to early Cardco memory boards
cinclude	cross-reference to C include files
mindwalker	clues to playing the game well
slideshow	make your own slideshows from the Kaleidoscope disk

AMICUS Disk 13
Amiga Basic programs
Routines from Carolyn Scheppner of CBM Tech Support, to read and display IFF pictures from Amiga Basic. With documentation. Also included is a program to do screen prints in Amiga Basic, and the newest BMAP files, with a corrected ConvertFD program. With example pictures, and the SaveILBM screen capture program.

Routines to load and play FutureSound and IFF sound files from Amiga Basic, by John Foust for Applied Visions. With documentation and C and assembler source for writing your own libraries, and interfacing C to assembler in libraries. With example sound.

Executable programs

gravity	Sci Amer Jan 86 gravitation graphic simulation, S-E-D

Texts

MIDI	make your own MIDI instrument interface, with documentation and a hi-res schematic picture.

AMICUS Disk 14
Several programs from Amazing Computing issues:
Tools

Dan Kary's	C structure index program, S-E-D

Amiga Basic programs:

BMAP Reader	by Tim Jones
IFFBrush2BOB	by Mike Swinger
AutoRequester	example
DOSHelper	Windowed help system for CLI commands, S-E-D
PETrans	translates PET ASCII files to ASCII files, S-E-D
C Squared	Graphics program from Scientific American, Sept 86, S-E-D
crlf	adds or removes carriage returns from files, S-E-D
dpdecode	decrypts Deluxe Paint, remo
ves copy	protection, E-D
queryWB	asks Yes or No from the user returns exit code, S-E
vc	VisiCalc type spreadsheet, no mouse control, E-D
view	views text files with window and
slider	gadget, E-D
Oing, Sproing, ya Boing style	Boing, Zoing are sprite-based demos, S-E-D

CLIClock, sClock, wClock are window border clocks, S-E-D

Texts
An article on long-persistance phospor monitors, tips on making brushes of odd shapes in Deluxe Paint, and recommendations on icon interfaces from Commodore-Amiga.

AMICUS Disk 15
The C programs include:

'pr'	a file printing utility, which can print files in the background, and with line numbers and control character filtering.
'fm'	displays a chart of the blocks allocated on a disk.
'Ask'	questions an 'execute' file, returns an error code to control the execution in that batch file
'Stat'	an enhanced version of AmigaDOS 'status' command.
'Dissolve'	random-dot dissolve demo displays IFF picture slowly, dot by dot, in a random fashion.
'PopCLI2'	invoke new CLI window at the press of a key.

The executable programs include:

'Form'	file formatting program through the printer driver to select print styles
'DiskCat'	catalogs disks, maintains, sorts,merges lists of disk files
'PSound'	SunRize Industries' sampled sound editor & recorder
'Iconmaker'	makes icons for most programs
'Fractals'	draws great fractal seascapes and mountain scapes.
'3D Breakout'	3D glasses, create breakout in a new dimension
'AmigaMonitor'	displays lists of open files,
memory use,	tasks, devices and ports in use.
'Cosmoroids'	version of 'asteroids' for the Amiga.
'Sizzlers'	high resolution graphics demo written in Modula 2.

Texts:

'ansi.txt'	explains escape sequences the CON: device responds to.
'FKey'	includes template for making paper to sit in the tray at the top of the Amiga keyboard.
'Spawn'	programmer's document from Commodore

Amiga, describs ways to use the Amiga's multitasking capabilities in your own programs.

AmigaBasic programs:

'Grids'	draw sound waveforms, and hear them played.
'Light'	a version of the Tron light-cycle video game.
'MigaSol'	a game of solitaire.
'Stats'	program to calculate batting averages
'Money'	"try to grab all the bags of money that you can."

AMICUS 15 also includes two beautiful IFF pictures, of the enemy walkers from the ice planet in Star Wars, and a picture of a cheetah.

AMICUS Disk 16

'juggler'	demo by Eric Graham. a robot juggler bouncing three mirrored balls, with sound effects. Twenty-four frames of HAM animation are flipped quickly to produce this image. You control the speed of the juggling. The author's documentation hints that this program might someday be available as a product.

IFF pictures
parodies of the covers of Amiga World and Amazing Computing magazines.

C programs:

'Inputhandler'	example of making an input handler.
'FileZap3'	binary file editing program
'ShowPrint'	displays IFF picture, and prints it.
'Gen'	program indexes and retrieves C structures and variables declared in the Amiga include file system.

Executable Programs:

'FixHunk2'	repairs an executable program file for expanded memory
'ms2smus'	converts Music Studio files to IFF standard 'SMUS' format. I have heard this program might have a few bugs, especially in regards to very long songs, but it works in most cases.
'Missile'	Amiga version of the 'Missile Command' video game,

This disk also contains several files of scenarios for Amiga Flight Simulator II. By putting one of these seven files on a blank disk, and inserting it in the drive after performing a special command in this game, a number of interesting locations are preset into the Flight Simulator program. For example, one scenario places your plane on Alcatraz, while another puts you in Central Park

AMICUS Disk 17
Telcommunications disk whuch contains six terminal programs.

"Comm" V1.33	term prog. with Xmodem, WXmodem,
"ATerm" V7.2	term prog. includes Super Kermit
"VT-100"V2.6	Dave Wecker's VT-100 emulator with Xmodem,Kermit, and scripting
"Amiga Kermit"	V4D(060) port of the Unix C-Kermit
"VTek" V2.3.1	Tektronix graphics terminal emulator based on the VT-100 prog. V2.3 and contains latest 'arc' file compression
"AmigaHost"	V0.9 for Compuserve. Includes RLE graphics abilities & CIS-B file transfer protocol.
"FixHunk"	expansion memory necessity
"FixObj"	removes garbage characters from modem received files
"Txt"	filters text files from other systems to be read by the Amiga E.C.
"addmem"	executeable version for use with mem expansion article in AC v2..1
'arc'	file documentation and a basic tutorial on un 'arc'ing files
"arcre"	for makeing "arc" files E.C.

AMICUS Disk 18

Logo	Amiga version of the populatcomputer language, with example programs, E-D
Tv"Text	Demo version of the TV"Text character generator
PageSetter	Freely distributable versions of the updated PagePrint and PageIFF programs for the PageSetter desktop publishing package.
FullWindow	Resizes any CLI window using only CLI commands, E-D
Life3d	3-D version of Conway's LIFE program, E-D
Defdisk	CLI utility to re-assign a new Workbench disk, S-E-D
Calendar.WKS	Lotus-compatible worksheet that makes calendars
SetKey	Demo of keyboard key re-programmer, with IFF picture to make function key labels, E-D
VPG	Video pattern generator for aligning monitors, E-D
HP-10C	Hewlett-Packard-like calculator, E-D
SetPrefs	Change the Preferences settings on the fly, in C, S-E-D
StarProbe	Program studies stellar evolution. C source included for Amiga and MS-DOS, S-E-D
ROT	C version of Colin French's AmigaBasic ROT program from Amazing Computing. ROT edits and displays polygons to create three dimensional objects. Up to 24 frames of animation can be created and displayed. E-D
Scat	Like Ing, windows on screen run away from the mouse, E-D
DK	Decays" the CLI window into dust, in Modula 2, S-E-D
DropShadow2	Adds layered shadows to Workbench windows, E-D

AMICUS Disk 19
This disk carries several programs from Amazing Computing. The IFF pictures on this disk include the Amiga Wake part T-shirt logo, a sixteen-color hi-res image of Andy Griffith, and five Amiga Live! pictures from the Amazing Stories episode that featured the Amiga.

Solve	Linear equation solver in assembly language, S-E-D
Gadgets	Bryan Catley's AmigaBasicGadTorial,
Household	Bryan Catley's AmigaBasic household inventory program, S-D
Waveform	Jim Shields' Waveform WorkAmigaBasic, S-D
DiskLib	John Kennan's AmigaBasic disk librarian program, S-D
Subscripts	Ivan Smith's AmigaBasic subscript example, S-D
String, Boolean	C programs and executables for Harriet Maybeck Tolly's Intuition tutorials, S-E-D
Skinny C	Bob Riemersma's example for making small C programs, S-E-D
COMAL.h	Make C look like COMAL header file,
EmacsKey	Makes Emacs function key definitions by Greg Douglas, S-D
AMon 1.1	Snoop on system resource use, E-D
BTE	Bard's Tale character editor, E-D
Size	CLI program shows the size of a given set of files, E-D
WinSize	CLI window utility resizes current window, S-E-D

AMICUS Disk 20

Compactor, Decoder	Steve Michel AmigaBasic tools, S-D
BobEd	BOB and sprite editor written in C,S-E-D
SpriteMasterII	Sprite editor and animator by Brad Kiefer, E-D
BlitLab	Blitter chip exploration C program by Tomas Rokicki, S-E-D
FPic	Image processing program by Bob Bush loads and saves IFF images, changes them with several techniques, E-D
Bankn	Complete home banking program, balance your checkbook! E-D

AMICUS Disk 21

Target	Makes each mouse click sound like a gunshot, S-E-D
Sand	Simple game of sand that follows the mouse pointer, E-D
PropGadget	Harriet Maybeck Tolly's proportional gadget example, S-E
EHB	Checks to see if you have extra-half-bright graphics, S-E-D
Piano	Simple piano sound program
CelScripts	Makes cel animation scripts for Aegis Animator, in AmigaBasic

This disk has electronic catalogs for AMICUS disks 1 to 20 and Fish disks 1 to 80. They are viewed with the DiskCat program, included here.

AMICUS Disk 22

Cycles	Light cycle game, E-D
Show_PrintII	Views and prints IFF pictures, including larger than screen
PrtDrvGen2.3	Latest version of a printer driver generator
Animations	VideoScape animations of planes and boing ball
Garden	Makes fractal gardenscapes
BasicSorts	Examples of binary search and insertion sort in AmigaBasic

AMICUS Disk 23
An AMICUS disk completely dedicated to music on the Amiga. This disk contains two music players, songs, instruments, and players to bring the thrill of playing "Big Sound" on your Amiga

Instruments	a collection of 25 instruments for playing and creating music. The collection ranges from Cannon to Marimba
List INSTR	program to list the instruments DMCS will not load as well as list the origins for any instrument.
Music	a collection of 14 Classical pieces
1812Overture	The 16 minute classical feature complete with Cannon!

Three Amiga Music Players:
SMUSPlay
MusicCraft2SMUS
MusicStudio2SMUS

AMICUS Disk 24

Sectorama	A disk sector editor for any AmigaDOS file-structured device, recover files from a trashed hard disk. By David Joiner of MicroIllusions
Iconize	Reduces the size of IFF images, companion program, Recolor, remaps the palette colors of one picture to use the palette colors of another. Using these programs and a tool to convert IFF brushes to Workbench icons, make icons look like miniatures of the pictures.
CodeDemo	Modula-2 program converts assembler object files to inline CODE statements. Comes with a screen scrolling example
AmiBug	Workbench hack makes the same fly walk across the screen at random intervals. Otherwise, completely harmless.
BNTools	Three examples of assembly language code from Bryce Nesbitt: 1. SetLace.prog to switch interlace on&off. 2. Why, replace AmigaDOS CLI Why 3. Loadit, prog to load a file into memory until a reboot. (Only the most esoteric hackers will find Loadit useful.)
Monolace	CLI program resets Preferences to several colors of monochrome & interlace screens. C source is included, works with DisplayPref, a CLI program which displays the current Preferences settings.
BoingMachine	A ray-traced animation of a perpetual motion Boing-making machine, includes the latest version of the Movie program, which has the ability to play sounds along with the animation. By Ken Offer
Daisy	Example of using the translator and narrator devices to make the Amiga talk. It is written in C.
QuickFlix	Script-driven animation and slideshow program flips through IFF images.
BMon	System monitor AmigaBasic program ; perform simple manipulations of memory.
Moose	Random background program, a small window opens with a moose resembeing Bullwinklesaying witty phrases user definable.
DGCS	Deluxe Grocery Construction Set, simple Intuition-based prog for assembling and printing a grocery list.
The Virus Check	directory holds several programs relating to the software virus that came to the US from pirates in Europe as detailed in Amazing Computing V2.12. Bill Koester's full explanation of the virus code is included. One program checks for the software virus on a Workbench disk; the second program checks for the virus in memory, which could infect other disks.

AMICUS Disk 25

Nemesis	Graphics demo pans through space towards the mythical dark twin of the sun with wonderful music and space graphics.
The KickPlay	directory holds text that describes several patches to the Kickstart disk. For Amiga 1000 hackers who feel comfortable patching a disk in hexadecimal, KickPlay offers the chance to automatically do an ADDMEM for old expansion memory, as well as the ability to change the picture of the "Insert Workbench" hand. A program is also included for restoring the correct checksum of the Kickstart disk.
KeyBird	BASIC prog edits keymaps, adjust the Workbench keymaps or create your own.

For PDS orders, please use form on page CIII

8ColorWB	Modifies the Workbench so three bitplanes are used, icons can have eight colors, instead of four, eight-color icons are included. Public domain program "zapicon" or "brush2icon" converts eight-color IFF brushes to icons, to use Deluxe Paint to make icons for this new Workbench.
BrushIcon	Converts brushes to icons (bizarr docs).
Egraph	Graphing prog reads [x,y] values from a file and displays them on the screen, similar to the same-named Unix program.
Keep 1.1	Message-managing program for telecommunications, lets you save messages from an online transcript to another file, understands the message format of the national networks and several types of bulletin board software. Moves through the transcript and save messages
Kill.fastdir	Speed up directory access, it creates a small file in each directory on a disk which contains the information about the files, will also remove all the "fastdir" files from each directory. by CLImate's authors
The LaceWB program	changes between interlace and non-interlace Workbench. Previously, you were forced to reboot after changing Preferences to an interlaced screen. This program flips between the normal and extended screen heights.
PW_Utility	A shareware utility for ProWrite users, changes margin settings and font types.
Guru	A CLI program, prints out probable causes for Guru mediations; C source included.
DiskWipe	Latest from Software Distillery, removes files from directories or disk drives, much faster than "delete."
Snow	AmigaBasic makes snowflake designs.
Mlist	Mailing list database.
Softballstats	Maintain softball statistics/ team records.
Dodge	Short Modula-2 program moves the Workbench screen around after a period of time, prevents monitor burn-in.

AMICUS Disk 26

Todor Fay's SoundScape	module code from his Amazing Computing articles. The source to Echo, Chord, TX, and VU is included. The Lattice and Manx C source code is here, along with the executable modules.
Claz2	Update of prog to convert IFF images to PostScript files for printing on laser printers
SDBackup	Hard disk backup prog with Lempel-Ziv compression to reduce the necessary number of disks.
TCB	Prints information about tasks and processes in the system; assembler source is included.
FunBut	Lets a function key act like a rapid series of left mouse button events.
DC	A handy program for people who use an Amiga 1020 5 1/4 inch drive as an AmigaDOS floppy.
	A Workbench program that sends a DiskChange signal to the operating system: instead of typing "diskchange df2:" over and over again, just click on the icon. C source included.
System config	File makes screen 80 columns wide of text in the Scribble! word processor.
Dick2Ram	2 programs to move the Scribble! spelling dictionary to and from the RAM disk.
Lexical	Analyzes a text file and gives the Gunning-Fog, Flesch, and Kincaid indices which measure readability.
HexDump	Modula-2 program to display memory locations in hexadecimal.
Tartan	AmigaBasic; design Tartan plaids.
DirMaster	Disk catalog program.
BMP	plays 8SVX sampled sounds in the background while something else is happening in the Amiga, as your Amiga is booting, for example.
ShowPt	CLI program changes your pointer to a given pointer.

AMICUS 26 also has a collection of mouse pointers, & Workbench program to display them

Fred Fish Public Domain Software

Fred Fish Disk 1:

amigademo	Graphical benchmark for comparing amigas.
amigaterm	simple communications program with Xmodem
balls	simulation of the "kinetic thingy" with balls on strings
colorful	Shows off use of hold-and-modify mode.
dhrystone	Dhrystone benchmark program.
dotty	Source to the "dotty window" demo on the Workbench disk.
treedraw	A small "paint" type program with lines, boxes, etc.
gad	John Draper's Gadget tutorial program
gfxmem	Graphical memory usage display prog.
halfbrite	demonstrates "Extra-Half-Brite" mode, if you have it
hello	simple window demo
lattfp	accessing the Motorola Fast Floating Point library from C
palette	Sample prog. to design color palettes.
trackdisk	Demonstrates use of the trackdisk driver.
requesters	John Draper's requester tutorial and example program.
speech	Sample speech demo program.
	Stripped down "speechtoy".
speechtoy	Another speech demo program.

Fred Fish Disk 2:

alib	Object module librarian.
cc	Unix-like frontend for Lattice C compiler.
dbug	Macro based C debugging package. Machine independent.
make	Subset of Unix make command.
make2	Another make subset command.
microemacs	Small version of emacs editor, with macros, no extensions
portar	Portable file archiver.
xrf	DECUS C cross reference utility.

Fred Fish Disk 3:

gothic	Gothic font banner printer.
roff	A "roff" type text formatter.
ff	A very fast text formatter
cforth	A highly portable forth implementation. Lots of goodies.
xlisp	Xlisp 1.4, not working correctly.

Fred Fish Disk 4:

banner	Prints horizontal banner
bgrep	A Boyer-Moore grep-like utility
bison	CNU Unix replacement 'yacc', not working.
bm	Another Boyer-Moore grep-like utility
grep	DECUS grep
kermit	simple portable Kermit with no connect mode.
MyCLI	Replacement CLI for the Amiga. V. 1.0
mandel	A Mandelbrot set program, by Robert French and RJ Mical

Fred Fish Disk 5

cons	Console device demo program with supporting macro routines.
freemap	Creates a visual diagram of free memory
input.dev	sample input handler, t raps key or mouse events
joystick	Shows how to set up the gameport device as a joystick.
keyboard	demonstrates direct communications with the keyboard.
layers	Shows use of the layers library
mandelbrot	IFF Mandelbrot program
mouse	hooks up mouse to right joystick port
one.window	console window demo
parallel	Demonstrates access to the parallel port.
printer	opening and using the printer, does a screen dump, not working
print.support	Printer support routines, not working.
proctest	sample process creation code, not working
region	demos split drawing regions
samplefont	sample font with info on creating your own
serial	Demos the serial port
singlePlayfield	Creates 320 x 200 playfield
speechtoy	latest version of cute speech demo
speech.demo	simplified version of speechtoy, with IO requests
text.demo	displays available fonts
timer	demos timer.device use
trackdisk	demos trakcdisk driver

Fred Fish Disk 6:

compress	like Unix compress, a file squeezer
dadc	analog clock impersonator
microemacs	upgraded version of microemacs from disk 2
mult	removes multiple occuring lines in files
scales	demos using sound and audio functions
setparallel	Allows changing parallel port parameters
setserial	Allows changing serial port parameters.
sortc	quicksort based sort program, in C
stripc	Strips comments and extra whitespace from C source

Fred Fish Disk 7:

This disk contains the executables of the game Hack V 1.0.1.

Fred Fish Disk 8:

This disk contains the C source to Hack on disk 7.

Fred Fish Disk 9:

moire	Draws moire patterns in black and white
MVP-FORTH	Mountain View Press Forth, version 1.00.03A. A shareware version of FORTH from Fantasia Systems.
proff	a more powerful text formatting program
setlace	Prog to toggle interlace mode on and off
skewb	a rubic's cube type demo
sparks	moving snake Graphics demo

Fred Fish Disk 10:

conquest	An interstellar adventure simulation game
dehex	convert a hex file to binary
filezap	Patch program for any type of file.
fixobj	Strip garbage off Xmodem transferred files.
iff	Routines to read and write iff format files.
ld	simple directory program
ls	Minimal UNIX ls, with Unix-style wildcarding, in C
sq,usq	file squeeze and unsqueeze
trek73	Star Trek game
yachtc	Dice game.

Fred Fish Disk 11:

dpslide	slide show program for displaying IFF images with miscellaneous pictures

Fred Fish Disk 12:

amiga3d	Shows a rotating 3 dimensional solid "Amiga sign".
ArgoTerm	a terminal emulator program, written in assembler
arrow3d	Shows a rotating 3 dimensional wire frame arrow.
ld4	directory listing program
IconExec	
SetWindow	two progs for launching progs from Workbench, presently only works under CLI.
SetAlternate	Makes an icon show a second image when clicked once
StarTerm	terminal emulator, with ASCII Xmodem, dialer, more.

Fred Fish Disk 13:

A Bundle of Basic programs, including:

Jpad	toybox	ezspeak	mandlebrot
xmodem	3dsolids	addbook	algebra
ror	amgseq1	amiga-copy	band
bounce	box	brickout	canvas
cardfi	circle	colorcircles	Copy
cubes1	cutpaste	datedogstar	
dragon	draw	dynamictriangle	
Eliza	ezterm	fillibuster	fractal
fscape	gomoku	dart	haiku
hal9000	halley	hauntedM	hidden
join	loz	mandel	menu
minipaint	mouse	Othello	patch
pena	pinwheel	gboxrandom-circles	
Readme	rgb	rgbtest	Rord
sabotage	salestalk	shades	shapes
shuttle			
sketchpad	spaceart	speakspeach	
speecheasy	spell	sphere	
spiral	striper	superpad	suprshr
talk	terminal		
termtest	tom	topography	triangle
wheels	xenos	xmostriper	

(note: some programs are Abasic, most are Amigabasic, and some programs are presented in both languages)

Fred Fish Disk 14:

amiga3d	update of #12, includes C source to a full hidden surface removal and 3D graphics
beep	Source for a function that generates a beep sound
dex	extracts text from within C source files
dimensions	demonstrates N dimensional graphics
filezap	update of disk 10, a file patch utility
gfxmem	update of disk 1, graphic memory usage indicator
gi	converts IFF brush files to Image struct, in C text.
pdterm	simple ANSI VT100 terminal emulator, in 80 x 25 screen
shell	simple Unix 'csh' style shell
termcap	mostly Unix compatible 'termcap' implementation.

Fred Fish Disk 15:

Blobs	graphics demo, like Unix 'worms'
Clock	simple digital clock program for the title bar
Dazzle	An eight-fold symmetry dazzler program. Really pretty!
Fish	double buffered sequence cycle animation of a fish
Monopoly	A really nice monopoly game written in AbasiC.
OkidataDump	Okidata ML92 driver and WorkBench screen dump program.
Polydraw	A drawing program written in AbasiC.
Polyfractals	A fractal program written in AbasiC.

Fred Fish Disk 16:

A complete copy of the latest developer IFF disk

Fred Fish Disk 17:

The NewTek Digi-View video digitizer HAM demo disk

Fred Fish Disk 18:

AmigaDisplay	dumb terminal program with bell, selectable fonts
Ash	Prerelease C Shell-like shell program, history, loops, etc.
Browser	wanders a file tree, displays files, all with the mouse
MC68010	docs on upgrading your Amiga to use a MC68010
Multidim	rotate an N dimensional cube with a joystick
PigLatin	SAY command that talks in Pig Latin
Scrimper	Screen image printer
Xlisp1.6	source, docs, and execut for a Lisp interpret.

Fred Fish Disk 19:

BlackJack	text-oriented blackjack game
JayMinerSlides	Slides by Jay Miner, Amiga graphics chip designer, showing flowchart of the Amiga internals, in 640 x 400.
Keymap_Test	test program to test the key mapping routines
LockMon	Find unclosed file locks, for programs that don't clean up.

Fred Fish Disk 20:

AmigaToAtari	converts Amiga object code to Atari form
DiskSalv	program to recover files from a trashed AmigaDOS disk.
Hash	example of the AmigaDOS disk hashing function
Hd	Hex dump utility ala Computer Language magazine, April 86
MandelBrots	Mandelbrot contest winners
MultiTasking	Tutorial and examples for Exec level multitasking
Pack	strips whitespace from C source
PortHandler	sample Port-Handler program that performs. Shows BCPL environment
Random	Random number generator in assembly, f or C or assembler.
SetMouse2	sets the mouse port to right or left
SpeechTerm	terminal Emulator with speech capabilities, XModem
TxEd	Demo editor from Microsmith's Charlie Heath

Fred Fish Disk 21

This is a copy of Thomas Wilcox's Mandelbrot Set Explorer disk. Very good!

Fred Fish Disk 22

This disk contains two new "strains" of microemacs.

Lemacs	version 3.6 by Daniel Lawrence. For Unix V7, BSD 4.2, Amiga, MS-DOS, VMS. Uses Amiga function keys, status line, execute, startup files, more.
Pemacs	By Andy Poggio. New features include <ALT> keys as Meta keys, mouse support, higher priority, backup files, word wrap, function keys.

Fred Fish Disk 23

Disk of source for MicroEmacs, several versions for most popular operating systems on micros and mainframes. For people who want to port MicroEmacs to their favorite machine.

Fred Fish Disk 24:

Conques	interstaller adventure simulation game
Csh	update to shell on Disk 14, with built in commands,named variables substitution.
Modula-2	A pre-release version of the single pass Modula-2 compiler originally developed for MacIntosh at ETHZ. This code was transmitted to the AMIGA and is executed on the AMIGA with a special loader. Binary only.

Fred Fish Disk 25

Graphic Hack	A graphic version of the game on disks 7 and 8 This is the graphics-oriented Hack game by John Toebes. Only the executable is present.

Fred Fish Disk 26

UniHunk	Processes the Amiga "hunk" loadfiles. Collect code, data, and bss hunks together, allows individual specificatio of code, data, and bss origins, and generates binary file with format reminiscent of Unix "a.out" format. The output file can be easily processed by a separate program to produce Motorola "S-records" suitable for downloading to PROM programmer. By Eric Black.
C-kermit	Port of the Kermit file transfer program and server.
Ps	Display and set process priorities
Archx	Yet another program for bundling up text files and mailing or posting them as a single file unit.

Fred Fish Disk 27

ABdemos	Amiga Basic demos; Caroly Scheppner.
NewConvertFD	creates .bmaps from fd files.
BitPlanes	finds addresses of and writes to bitplanes of the screen's bitmap.
AboutBmaps	A tutorial on creation and use of bmaps.
LoadILBM	loads and displays IFF ILBM pics.
LoadACBM	loads and displays ACBM pics.
ScreenPrint	creates a demo screen and dumps it to a graphic printer.
Disassem	Simple 68000 disassembler. Reads standard Amiga object files and disassembles the code sections. Data sections are dumped in hex. The actual disassembler routines are set up to be callable from a user prog so instructions in memory can be disassembled dynamically. By Bill Rogers.
DvorakKeymap	Example of a keymap structure for the Dvorak keyboard layout. Untested but included because assembly examples are few and far between. By Robert Burns
Hypocycloids	Spirograph, from Feb. 84 Byte.
LinesDemo	Example of proportional gadgets to scroll a SuperBitMap.
MemExpansion	Schematics and directions for building your own homebrew 1 Mb memory expansion, by Michael Fellinger.
SafeMalloc	Program to debug 'malloc()' calls
ScienceDemos	Convert Julian to solar and sidereal time, stellar positions and radial velocity epoch calculations and Galilean satellite plotter. By David Eagle.

Fred Fish Disk 28

ABasic games by David Addison: Milestone, and	Backgammon , Cribbage, Othello
Cpp	DECUS 'cpp' C preprocessor, & a modified 'cc' that knows about the 'cpp', for Manx C.
Shar	Unix-compatible shell archiver, for packing files for travel.
SuperBitMap	Example of using a ScrollLayer, syncing SuperBitMaps for printing, and creating dummy RastPorts.

Fred Fish Disk 29

AegisDraw Demo	Demo program without save and no docs.
Animator Demo	Player for the Aegis Animator files
Cc	Unix-like front-end for Manx C
Enough	Tests for existence of system resources, files, and devices
Rubik	Animated Rubik's cube program
StringLib	
Vt100	VT-100 terminal emulator with Kermit and Xmodem protocols

Fred Fish Disk 30

Several shareware programs. The authors request a donation if you find their program useful, so they can write more software.

BBS	an Amiga Basic BBS by Ewan Grantham
FineArt	Amiga art
FontEditor	edit fonts, by Tim Robinson
MenuEditor	Create menus, save them as C source, by David Pehrson
StarTerm3.0	Very nice telecom by J. Nangano

(Fred Fish Disk#30 is free if requested when ordered with at least three other disks from the collection.)

Fred Fish Disk 31

Life	Life game, uses blitter to do 19.8 generations a second.
Mandelbrot	Version 3.0 of Robert French's program.
MxExample	Mutual exclusion gadget example.
RamSpeed	Measure relative RAM speed, chip and fast.
Set	Replacement for the Manx "set" command for environment variables,with improvements.
Tree	Draws a recursive tree, green leafy type, not files.
TxEd	Crippled demo version of Microsmith's text editor, TxEd.
VDraw	Full-featured drawing program by Stephen Vermeulen.
Xicon	Invokes CLI scripts from icon
Ticon	Displays text files from an icon.

Print1.2 CLI-based printing utility with several nice features including the ability to print in ASCII or HEX, with or without line numbers and a CTRL-C Trap. Includes source. Author: John F. Zacharias

Sh Another version of the 'Sh' utility to unshar shell archives. Apparently corrects some problems encountered by similar programs. Until we can get everybody using some sort of "standard", perhaps if we collect enough of these utilities, we will eventually find one that works with the particular archive we're trying to unshar! Includes source. Author: Jim Guilford

Strings A simple utility with command-line options for locating strings in a binary file. V1.0, includes source. Author: Joel Swank

TitlePage Prints banner-type title pages for identifying listings. Lots of command-line options for specifying various fonts, pitches, typestyles, selectable centering, etc. Includes source. Author: Joel Swank

Tunnel An interesting graphics demo written in TDI-Modula 2. I suggest you don't stare at this too long! Includes source. Author: Garth Thornton

Fred Fish Disk 175

Elements Very nice interactive display of the the Periodic Table of Elements. Can display a large amount of pertinent data about a selected element along with a good deal of general and miscellaneous info. Author: Paul Thomas Miller

Furnish For those of you who may have ever used the "scale size cut and place" method of determining your next living-room arrangement, this Amiga-ized version may be just what you need. Binary only, shareware. Author: Terry Gintz

Plot Program to compute and plot 3 dimensional functions. Major revision to PD version on FF49 . V4.1, binary only, shareware. By: Terry Gintz

SafeBoot Very handy intuition-based program to read and save custom bootblocks. The bootblock can then be later restored should the disk become virus-infected. V2.2, binary only. Author: Mark Lanoux

SendMorse Brush up on your morse code with this simple program that will read an input textfile and output the characters at an adjustable rate. By: Joe Larson

VirusX V 3.10 of the popular virus detection/vaccination program. Features a test for the new IRQ virus, among others, and a new "Kill Virus" utility. Includes source. Author: Steve Tibbett

WBDepth CLI program that allows you to change the number of bitplanes for the WB screen on the fly. Very useful for A500 and A2000 users with kickstart in ROM. Binary only. By: Andry Rachmat

Zippy A "Graphical Shell". Opens a medium-size window and attaches a menu-strip for performing all sort of disk/data manipulations. Features script files allowing you to attach custom menu selections as you move between directories. Also included is an intuition based utility for altering FileInfo data. (filename, filenote, RWED attributes, etc.) V 2.5, binary only. By: MWeiblen

Fred Fish Disk 176

AnalytiCalc V23-2A of Glenn Everhart's large and powerful spreadsheet program called AnalytiCalc, submitted to me directly by Glenn for inclusion in the library an update to FF144. AnalytiCalc is presented in entirely ZOOed form because it could not otherwise fit on a single disk. With this release, AnalytiCalc has become "Freeware" rather than "Shareware". Thus the only restrictions on AnalytiCalc code are that derivative programs remain freely distributable.

HyperNet HyperNet is a small hypertext shell program for Amiga, presented with sources and brief documents. . HyperNet allows a "master" AmigaDos process to control a series of connected processes, where the connections are randomly ordered directed graphs. Permissible "child" processes available at any stage are governed by the links of the graph. The implementation is mainly instructive, but can be used for tutorials or demonstrations and illustrates the simplicity of hypertext concepts on a multi-tasking system. Author Glenn Everhart

Fred Fish Disk 177

ASpice A version of the SPICE 2G.6 circuit analysis program which has been modified to run in the Amiga environment. The program arrays are adjusted to require one forth the memory of the DEC VAX version. Although this does not usually put much of a constraint on circuit analysis, some users who are used to the full mainframe environment may have to be more aware of the memory demands of their analysis. Requires a minimum of 1.5 MB memory. This version neither supports nor requires the 68020 processor or 68881 coprocessor. Binary only. By Many

DiskSalv V1.32 of the popular "undelete" and file recovery program. Fixes a few bugs apparently found on the V1.3 on FF164. Author: Dave Haynie

Jask An intuition-based replacement for the ASK command. Will bring up a requester with a message and boolean yes/no type gadgets. V 1.0, includes source. Author: John Barshinger

Marge A simple CLI utility to add a specified number of spaces or tabs to the left side of every line in a file. Includes source. Author: Joel Swank

Path An interesting concept in path-searching. This program contains a path-handler that allows you to selectively control or assign your system's search path using script files. Includes source. By: Rico Mariani

Fred Fish Disk 178

AmicForm Creates a phonebook containing only those areacodes and exchanges reachable through PC-Pursuit. Update to FF157. Works with the new Finalist BBS format. V1.4, Binary only. By: John Motsinger

BlackBox The black box is an 8x8 grid in which several "atoms" are hidden. Your job is to find the atoms. You have at your disposal a ray projector which you can use to send rays into the box from any of the 32 spots around the box. Binary only. Author: Tim Kemp

ClATimer Two versions of clatimer routines to provide precise timing for applications requiring a high-accuracy real-time clock. Includes source and a sample executable. Author: Karl Lehenbauer, based on the original version by Paul Higginbottom

Cosmic An interstellar multiplayer game of War and Peace. From the looks of the documentation file, it appears fairly extensive! V1.01, includes source. By: Carl Edman

Ls V2.0 of the popular UNIX style directory lister. Revised for Lattice 5.0 and made 1.3 compatible. Includes source. Author: Justin V. McCormick.

RemLib Removes a specified library (if currently unused) or displays some information about all available libraries. Update to FF139. V1.11, includes source in assembler. Author: Heiko Rath

RexxArpLib V2.0 of the rexxarplib.library, which has grown considerably, with substantial intuition interface support. Also included is a large number of ARexx macros. Author: W.G.J. Langeveld

Fred Fish Disk 179

DietAid Diet planning aid to allow the user to compile lists of ingredients (recipes) and automatically compute calorie totals, etc. Update FF36. V3.1, binary only. by Terry Gintz

Dmake Beta release of Matt's version of the UNIX make utility. Features multiple dependancies, wildcard support, and more. Includes source. By Matt Dillion

Exception Exception is a set of error handling routines that provide a programmer with the ability to easily handle often difficult to implement routines. Routines such as no more memory, file not open, read/write error....etc. V0.6, includes source. By Gerald T Hewes

KickFont For A-1000 owners, will permanently replace the topaz font on the kickstart disk with a font called "look". Includes a sample in the form of an IFF picture. V3.0, binary only. Also included is Benjamin Fuller's freely redistributable 'SumKick' program. By Greg Browne

Launch Sample program showing how you can load and execute a program in the workbench environment, then return to the CLI. Includes source. By Peter da Silva

Regexp A nearly-public-domain reimplementation of the V8 regexp(3) package. Gives C programs the ability to use egrep-style regular expressions, and does it in a much cleaner fashion than the analogous routines in SysV. Includes source. By Henry Spencer

TSnip Very nice "cut and paste" type utility with lots of uses and functions. Features a pop-up intuition control panel, multiple font and color recognition, clipboard and pipe support and a couple of utility programs. V1.4a, source for support programs only. By John Russell

UnixUtil A few CLI utilities, including some functionally similiar to the UNIX utilities of the same names. Included are: Wc, Head, Tail, Tee, Detab, Entab, and Trunc. Descriptions are given in the included '.doc' files. By Gary Brant

Fred Fish Disk180

Browser A programmer's "Workbench". Allows you to easily and conveniently move, copy, rename, and delete files & directories from a CLI environment. Also provides a method to execute either Workbench or CLI programs. V1.6, update to FF134, binary only. By Peter da Silva

GeoTime A couple of interesting "clock" type programs based on the "Geochron". Observe the earth's shadow scroll across a map or globe in real-time, based on the system clock. V1.0, binary only, shareware. By Mike Smithwick

GPrint A black & white graphics print utility for Epson compatible printers. Command-line options allow several different print qualities and densities. Includes a couple of sample IFF files for printing. V2.03, binary only, shareware. By Peter Cherna

Jed A nicely done, intuition-based editor that is quite user-friendly. Features word-wrap, auto-indent, newcli, alt buffer, split-window, keyboard macro, help, printing, and more. V1.0, binary only, shareware. By Dan Burris

NoVirus Another Anti-Virus utility. This one features known and new virus detection, view boot block, save and restore bootblocks, several "install" options and more. Written in assembly. V1.56, binary only. By Nic Wilson

RepString Nice little CLI utility to replace any type of string in any type of file with another string of any type. V1.0, binary only, shareware. By Luciano Bertato

TrekTrivia Very nice mouse-driven trivia type program for Star Trek fans. Contains 100 questions with additional trivia disks available from the author. Binary only, shareware. By George Broussard

Fred Fish Disk 181

AMXLISP Amiga-ized version of the XLisp interpreter originally by David Betz. V2.00, includes source. By David Betz; Amiga work by Francois Rouaix

Bally Amiga port of the former arcade game named Click. Lacks sound effects,promised for later updates. V0.1, binary only, shareware. By Oliver Wagner

Tracker Useful debugging routines similar in function but more versatile to those of "MemTrace" on FF163. Will track and report on calls to AllocMem(), FreeMem() [or lack thereof!] among others. V0.0a (Alpha release). By Karl Lehenbauer

Fred Fish Disk 182

AMC "Amiga Message Center". Scrolls a message from a text file across the screen on a colorful background. Similar to the "greetings" programs developed by European Amiga enthusiasts. V1.0, binary only. By Foster Hall

Edimap A keymap editor. Allows you to read in an existing keymap file, modify it to suit your needs, and save it as a reacy-to-use keymap. V1.0, includes source. Author: Gilles Gamesh

HR136 An IFF file containing a chart showing every possible mixture of the sixteen basic palette colors. Also included are optimized and monochrome palettes along with several tips and techniques for using them with various paint programs.By Dick Bourne

Iconmerger Intuition-based program to take any two brush files and merge them into an alternate-image type icon. V2.0, binary only. By Terry Gintz

Sam Another IFF sound player with several command-line options. Includes several samples. V1.0, binary only. By Nic Wilson

SetFont Allows you to change the system font with various command-line options. Cleans up all known bugs in FF75. V2.5, includes source in C++. By Dave Haynie

Fred Fish Disk 183

FixFd A utility for Amiga assembly programmers. FixFd will read a '.FD' file and output a file that can be 'INCLUDE'ed rather than having to link with the colossal 'Amiga.Lib'. V1.0, includes source in assembly. By Peter Wysplanski

Mklib Another example of building a shared library that evolved from "Elib" FF87. Also included is a library, Edlib, which contains several functions not included in the Manx standard libraries. Includes source. By Edwin Hoogerbeets with C-functions from several different authors

PCQ A subset implementation of a freely-redistributable Pascal compiler. Supports include files, external references, records, enumerated types, pointers, arrays, strings and more. Presently does not support range types, the 'with' statement or sets. V1.0, includes source and sample programs. By Patrick Quaid

Fred Fish Disk 184

BI A small brush to to C-code image converter, intended to be used from CLI. V1.0, binary only. By Terry Gintz

CardMaker A programmer's aid for creating card image data that can be used in any card game that uses the standard 52 card deck. V1.0, binary only. By Terry Gintz

DPS Demo version of a program that will allow you to take any IFF file and save it as a totally self-contained executable file, without the need for any IFF-viewers. V1.0, binary only. By Foster Hall

MouseUtil Intuition based program to allow you to change your mouse speed without having to go through preferences. V1.1, includes assembly source. By Luciano Bertato

Print Small print utility designed to replace the "copy <filename> to prt:" command. Opens a window displaying the filename being printed, length, and a status bar showing percent completed. Also includes an abort gadget. V1.0, binary only. By Luciano Bertato

VacBench This amusing little screen hack will "clean up" your WorkBench screen for you when it gets too cluttered! Binary only. By Randy Jouett

World A text adventure game similar to the Infocom adventures of Planetfall and Starcross. Quite large with a tremendous variety of responses. V1.02, includes source. By Doug Mcdonald, Amiga port by Eric Kennedy

Fred Fish Disk 185

Commodore IFF This is a copy of the official November 1988 Commodore IFF disk. All the files in the "documents" directory are in zoo file "documents.zoo"

Fred Fish Disk 186

A68k A 68000 assembler originally written in Modula-2 in 1985 and converted to C by Charlie Gibb in 1987. Has been converted to accept metacomco-compatible assembler source code and to generate Amiga objects. Includes source. This is V2.42, an update to FF110. By Brian Anderson; C translation and Amiga work by Charlie Gibb

Cards'O'Rama A simple game that let's you push your memory. It is played with a deck of 32 cards, grouped in 16 pairs. The cards are shuffled and then displayed at the beginning of each game. Your goal is to pick up as many pairs as you can, until there are no cards left on the screen. V1.0, includes source. By Werther Pirani

Qt2 A cute program that gives the time the way many people actually do, I.E. "it's nearly ten to five". Includes source in assembly. By Charlie Gibb

SimCPM A CP/M simulator for the Amiga. Simulates an 8080 along with H19 terminal emulation. Includes source. This is V2.3, an update to FF109. By Jim Cathey; Amiga port by Charlie Gibbs and Willi Kusche

Fred Fish Disk 187

Diskperf A disk benchmark program which runs on both Unix and the Amiga. This is an update to FF48, with bug fixes and more reliable measurements of the faster read and write speeds available under the new Fast File System. By Rick Spanbauer, enhancements by Joanne Dow

HackLite This is the latest version the Amiga port of Hack, with lots of Amiga specific enhancements and neat graphics. Now includes an easy to use installation program. This is HackLite V1.0.0, binary only. By Software Distillery

Mackie A versatile cli/macro-key initiator based on POPCLI with a unique method of "screen-blanking". I won't say more, just try it! V 1.13, includes source. This is an update to FF161. By Tomas Rokicki

SetCPU A program designed to allow the user to detect and modify various parameters related to 32 bit CPUs. Includes commands to enable or disable the text/data caches, switch on or off the '030 burst cache line fill request, use the MMU to run a ROM image from 32-bit memory, and to report various parameters when called from a script. V1.4, includes source. By Dave Haynie

Fred Fish Disk 188

BootIntro This program creates a small intro on the bootblock of any disk, which will appear after you insert the disk for booting. The headline can be up to 20 characters. The scrolling text portion can be up to 225 characters. V1.0, binary only. By Roger Fischlin

DiffDir Diffdir compares the contents of two directories, reporting on differences such as files present in only one directory, different modification dates, file flags, sizes, comments, etc. V1.0, includes source. By Mark Rinfret

ExecDis A disassembler comment generator program for the 1.2 Kickstart ROM exec library image. Generates a commented disassembly of the exec library. V1.0, binary only. By Markus Wandel

FastGro A fractal program, simulating Diffusion-Limited Aggregation (DLA) as described in the December 1988 Scientific American in the Computer Recreations column. This program is about an order of magnitude faster than the "SLO GRO" program described in Scientific American. V 1.0, includes source. By Doug Houck

FracGen A fractal generator program that generates fractal pictures from "seeds" that you create. This is unlike any of the other "fractal generators" I've seen. It can be used to load and display previously created fractal pictures, modify existing fractals, or create your own fractals. V 1.23, binary only, update to FF142. By Doug Houck

MemoryClock A clock program that shows the amount of free fast ram, free chip ram, as well as the time and date. Includes source in assembly code. By Roger Fischlin

MinRexx A simple ARexx interface which can be easily patched into almost any program. Includes as an example the freedraw program from FF1. Includes source. By Tomas Rokicki

Null A new dos device that behaves like "NIL:" but unlike "NIL:", it is a real handler. This makes it useful in lots of situations where "NIL:" cannot be used. V 0.0, includes source. By Gunnar Nordmark

TextDisplay A text display program, like "more" or "less", but about half the size and handles all screen formats (pal/ntsc, interlace/non-interlace, etc). V 1.1, binary only. By Roger Fischlin

Fred Fish Disk 189

Mackie A versatile cli/macro-key initiator based on POPCLI wit a unique method of "screen-blanking". I won't say more, just try it! Version 1.20, includes source. update of FF187. Author: Tomas Rokicki

NetHack This is part 1 of a two part distribution of NetHack, which was too large to fit on a single disk, even when zoo'd. Part 2 is on disk 190. Both parts, along with zoo to unpack them, are required to use or rebuild NetHack. V 2.3. Includes source. Author: Various; Amiga work by Olaf Seibert

Uedit V 2.4g shareware editor. Has learn mode, a command language, menu customization, and other user configurability and customizability features. Binary only, shareware, Update to FF 173 Author: Rick Stiles

Fred Fish Disk 190

GaryIcons A collection of more interesting and useful icons. Author: Gary Roseman

LBM2Image Takes an IFF picture and generates a C source module which can be compiled and linked with your program to display the picture with the intuition DrawImage function. Binary only. by: Denis Green

NetHack This is part 2 of a two part distribution of NetHack, which was too large to fit on a single disk, even when zoo'd. Part 1 is on disk 189. Both parts, along with zoo to unpack them, are required to use or rebuild NetHack. V 2.3 Includes source. Author: Various; Amiga work by Olaf Seibert

Fred Fish Disk 191

BlitLab Blitlab is a program which lets you experiment with the blitter, to your hearts content, in relative safety. It opens a workbench window with gadgets for all the registers of the blitter, and allows you to manipulate individual registers and perform blits on a magnified bitmap. V 1.4, an update to FF84. Includes source. Author: Tomas Rokicki

Blk A requester making tool employing various recursive algorithms including a recursive parser. It takes input text files and converts them to C-source for including as requester declarations. Update to FF152, with many enhancements. Includes source. Author: Stuart Ferguson

FileBootBlock This simple little program reads blocks 0 and 1 of a bootable disk and saves them as a program file that can be run (heaven forbid) or disassembled by programs like DiS or DSM. Includes source in assembly code. by: John Veldthuis

Spell A port of a Unix version of a freely distributable screen oriented, interactive, spelling checker. Update to FF54, with enhancements by Tomas Rokicki. V2.0.02, includes source. Author: Pace Willisson; enhancements by Tomas Rokicki

Pz15 Computer version of those cheap plastic puzzles with 15 white tiles numbered 1 through 15 and an empty square in a 4 by 4 arrangement. This one is more challenging since you can't solve it by just prying out the pieces. Includes source. Author: Mike Hall

Fred Fish Disk 192

Eval This package allows you to manipulate expressions. Currently its two main functions are evaluation and differentiation. It also does some basic simplifications (based on pattern matching) to make the result of a differentiation more presentable. Includes source. Author: David Gay

PacMan87 This is a nice little "pacman like" game with some new features like fire pits, stabbing knives, electric arcs and flame throwers, that must be avoided. Has three levels of difficulty, easy, medium, and hard. Sounds can be toggled on or off. Keeps a record of the top ten scores. Shareware, binary only. Author: Steve Jacobs and Jim Boyd

ReSourceDemo A demo version of ReSource, an interactive disassembler for the Amiga. This is a complete version except that the "save" features have been disabled. V0.36, binary only. by: Glen McDiarmid

For PDS orders, please use form on page CIII

Fred Fish Disk 193

KeyMapEd Allows you to change the KeyMaps used with SetMap This is a full featured editor providing support for normal, string and dead keys. The keyboard represented is from an A2000/A500 but it is fully compatible with A1000 keyboards. V1.02, includes source Author: Tim Friest

Zc This is a modified version of the Sozobon C compiler fromFF171. It has been modified to generate code compatible with the A68k assembler from FF186 and a new frontend control program makes it easy to use like the UNIX "cc" frontend. V1.01, includes source. by: Johann Ruegg; Amiga work by Joe Montgomery

Fred Fish Disk 194

Moria A single player dungeon simulation. The object of the game is to defeat the Balrog, which lurks in the deepest levels of the dungeon. You begin at the town level above the dungeon, where you may acquire supplies, weapons, armor, and magical devices by bartering with various shop owners, before descending into the dungeon to do battle. Amiga enhancements include pull down menus, graphics mode, pickup mode, a continuous move mode, a real time mode, a message wait time mode, as well as other modifications to improve overall playability and to take advantage of the unique features of the Amiga. V3.0, binary only, requires at least 1Mb of memory. Author: Robert Alan Koeneke and others. Amiga version by Richard Henderson and others.

Fred Fish Disk 195

MicroEMACS Version 3.10 of Daniel Lawrence's variant of Dave Conroy's microemacs. This is an update to the version released on disk 119. New features include multiple marks, more function key support, a better crypt algorithm, and end-of-word command, a command line switch for setting environment variables, new hooks for macros, a command to strip trailing whitespace, international-zation features like foreign language message support, horizontal window scrolling, much faster search algorithm, Amiga intuition support, and more. Includes source and extensive online documentation. Author: Dave Conroy, MANY enhancements by Daniel Lawrence

Fred Fish Disk 196

HamPics These are some of the most stunning digitized pictures yet for the Amiga. They were scanned at a resolution of 4096 by 2800 pixels, 36-bits per pixel, on an Eikonix 1435 slide scanner, cropped, gamma corrected, scaled, and converted to Amiga IFF HAM files. They are displayed with a special ILBM loader that handles overscan HAM images. Includes source for the display program. Author: Jonathan Hue

Fred Fish Disk 197

CTags Create a tags file from the specified C, Pascal, Fortran, YACC, lex, or lisp sources. A tags file can be used by a cooperating editor to quickly locate specified objects in a program's source code. Berkeley V4.7, includes source. Author: Ken Arnold, Jim Kleckner, and Bill Joy Ported to Amiga by G. R. (Fred) Walter

Find Find is a utility which searches for files that satisfy a given boolean expression of attributes, starting from a root pathname and searching recursively down through the hierarchy of the file system. Very much like the Unix find program. V1.2, includes source. Update to FF134 Author: Rodney Lewis

FixHunk A program to modify executable files to allow them to run in external memory. It forces all DATA and BSS hunks in the file to be loaded into CHIP memory. CODE hunks will still load into FAST ram if available. New features include an interactive mode to select where each DATA or BSS hunk will load into memory, support for overlays, support for AC BASIC compiled programs, and support for new hunk types as used by "blink". V2.1, binary only. Update to FF36. Author: D.J. James

Nro Another roff style text formatter. This is version 1.5, an update to the version released on disk 79. New features include generation of ANSI/ISO codes for bold, italics, and underline, more than one formatting command on a line, longer macro names, and many more formatting commands. Includes source. Author: Unknown, posted to usenet by Alan Vymetalik Many enhancements by Olaf Seibert

Stevie A public domain clone of the UNIX 'vi' editor. Supports window-sizing, arrow keys, and the help key. V3.35a, includes source. Update to FF166. Author: Various, Amiga work by G. R. (Fred) Walter

Fred Fish Disk 198

Charon Charon is Bradley's entry for the First Annual Badge Killer Demo Contest. The text of the demo was written by Lord Dunsany (long before the Amiga). Bradley created the illustrations and animation. The sound track is a traditional Scottish tune "The Arran Boat". by: Lord Dunsany (1915), Bradley Schenck (1988)

Fred Fish Disk 199

ASimplex An implementation of the Simplex algorithm for solving linear programs. It uses the standardized MPSX-format for input data files. V1.2, includes source. Author: Stefan Forster

Csh V3.02a of a csh like shell derived from Matt Dillon's shell, V2.07. Includes many new or improved commands, some bug fixes, etc. Includes source. Author: Matt Dillon, Steve Drew, Carlo Borreo, Cesare Dieni

MIDIsoft A program to transfer sound samples between the Amiga and a Roland S-220. V1.0, binary only. Author: Dieter Bruns

Pyro A screen blanking program that goes beyond the normal blanking process. When there are no input events, pyro takes over and starts a little fireworks display in color. V1.1, binary only. Author: Steve Jacobs and Jim Boyd

SnipDemo Demo version 1.23 of signal processing program sold by Digital Dynamics. Binary only. Author: John Hodgson

Viewer A very small program for displaying IFF pictures of any resolution. This one is written in assembly code and is only 988 bytes long. Binary only. Author: Mike McKittrick

Fred Fish Disk 200

NotBoingAgain Dr. Gandalf's entry for the First Annual Badge Killer Demo Contest. It is an interlaced HAM animation with nicely integrated sound effects. It is a great visual pun on the original Boing demo, but to say anymore would ruin the effect. Binary only, requires 1 Mb of memory. Author: Dr. Gandalf (Eric J. Fleischer, MD)

Tank This is Vincent's entry for the First Annual Badge Killer Demo Contest. It is an animation of a "fishtank simulator", with sound effects and a cute twist. Binary only. Author: Vincent H. Lee

Fred Fish Disk 201

Draco Update to Chris Gray's Draco distribution for the Amiga.Enhancements include support for floating point, register variables, more optimization, improved call/returnstandard, etc. V1.2, an update to FF76. Requires documentation from FF77 to complete the distribution kit. Binary only. Author: Chris Gray

DropCloth DropCloth lets you place a pattern, a 2 bitplane IFF image or a combination of a pattern and image, into the WorkBench backdrop. This is version 2.4, an update to version 2.2 on disk 128. Shareware, binary only. Author: Eric Lavitsky

Fred Fish Disk 202

SlavicFonts A whole bunch of new fonts from Robin LaPasha. Version 1.0. Author: Robin LaPasha

Vlt VLT is both a VT100 emulator and a Tektronix (4014 plus subset of 4105) emulator, currently in use at SLAC (Stanford Linear Accelerator Center). Although the VT100 part was originally based on Dave Wecker et al.'s VT100, many enhancements were made. The program requires ARP, and it has an ARexx port. XMODEM 1K/CRC and Kermit protocol support also included. Version 3.656, binary only. Author: Willy Langeveld

Fred Fish Disk 203

Examples Assembly and C code examples, including some old favorites (like speechtoy and yachtc3) downcoded to assembly language. Includes a replacement for the official audio device, an example of creating a subtask, a rewrite in assembly of R. J. Mical's file requester, an example of installing a custom input handler ahead of intuition, and more. Author: Jim Fiore & Jeff Glatt

GurusGuide The source files for all examples published in the "Guru's Guide, Meditation #1: Interrupts" by Carl Sassenrath, the architect of the Amiga's low-level multitasking operating system and designer of Exec. Author: Carl Sassenrath

Isam A library of routines to access relational data base systems using the Index Sequential Access Method (ISAM). This is beta version 0.9, binary only. Author: Kai Oliver Ploog

Fred Fish Disk 204

FileReq A simple file requestor, written as an exercise by the author to see how easy it would be (it wasn't). Includes source. Author: Jonathan Potter

GnuGrep The grep program from the GNU project. Replaces grep !grep, egrep, and bmgrep. Currently does not expand Amiga style wildcards, so if you wish to scan multiple files you will need to use it with a shell that does this for you. Version 1.3, includes source. Author: Mike Haertel, James Woods, Arthur Olson, Richard Stallman, Doug Gwyn, Scott Anderson, Henry Spencer

HAMCu Installs a custom copper list for the current active view (usually workbench) that contains all the colours from 0x000 to 0xfff. A neat effect and an easy way to show off the color capabilities of the Amiga. Includes source. Author: Jonathan Potter

Image-Ed An shareware icon editor submitted by the author for inclusion in the library. Suggested shareware donation of $20. Version 1.8, binary only. Author: Jonathan Potter

JPClock A short clock program that is just packed with features. Includes source. Author: Jonathan Potter

MouseBounce A short hack/game that makes your mouse pointer bounce around the screen. The object is to close the MouseBounce window and exit the game. Each time you click the mouse button, the pointer speeds up. Includes source. Author: Jonathan Potter

PopDir A small utility which "pops open" to help you look at the contents of a particular directory on demand. Version 1.4, includes source. Author: Jonathan Potter

PopInfo A small utility which "pops open" to give you information about the status of your devices and memory. Version 2.9, includes source. Author: Jonathan Potter

Teacher Teacher is a short, simple hack. I won't spoil the fun by telling you what it does. Includes source. Author: Jonathan Potter

Fred Fish Disk 205

Bally Amiga port of the former arcade game named Click. This version now has sound effects. Version II, an update to the version released on disk 181. Binary only, shareware. Author: Oliver Wagner

BattleForce A nicely done shareware game, submitted by the author, that simulates combat between two or more giant, robot-like machines. Binary only, version 3.01. Author: Ralph Reed

Chess A port of a chess game posted to Usenet. This is an update to the version first included on disk 96. It has been upgraded to use an Amiga Intuition interface. Version 2.0, binary only. Author: John Starback; ported to Amiga by Bob Leivian Version 2.0 upgrades by Alfred Kaufmann

Fred Fish Disk 206

Brownian A demo based on both fractal theory and brownian motion. Includes source. Author: John M. Olsen

Hawk A stereo image of a hawk. Requires red/green stereo glasses to view. Author: Unknown (no documentation included)

MemFlick Treats all the memory in your Amiga like it was part of a bitplane inside a graphics display. Provides sort of a graphical picture of your memory usage. Binary only. Author: Jim Webster

PeX A demo of the various graphics capabilities of the Amiga. Author: Unknown (no documentation included) PictureGarden Another demo, apparently in compiled basic. Author: Unknown (no documentation included)

StereoDemo A demo of stereoscopic graphics, written in assembly language. Requires red/green stereo glasses to view. Includes sources. Author: David M. McKinstry

Triple Three demos of some of the Amiga's graphics and sound capabilities. Binary only. Author: Tomas Rokicki

Fred Fish Disk 207

Coyote Gene's entry to the 1988 Badge Killer Demo contest. A very cute (and large) animation. Requires about 1900 blocks of disk space, so it is distributed in "arc format". Author: Gene Brawn

Fred Fish Disk 208

AsteroidField This is Michael's entry for the 1988 Badge Killer Demo Contest. It is a large animation of a spacecraft flying madly through an asteroid field (chased by unseen foes) that includes a couple of near misses. Author: Michael Powell

Users Groups !

We want to here from you.

Amazing Computing **understands that the momentum and excitement behind the Amiga is generated at the Users Group level. Users Groups are the backbone of information exchange among Amiga users.**

AC wants to maintain the most complete and up to date list of Amiga Users Groups and BBSs in the world. We publish these lists in our product guides, *AC's Guide to the Amiga*, and will be printing the updated lists in an upcoming issue. Information received before July 6 will be printed in the fall *AC's Guide to the Amiga*.

If you would like your group to be listed, please send the following information:

- **Group name.**
- **Address and phone number.**
- **Point of contact.**
- **Meeting dates, time and place.**
- **Your group's BBS and BBSs your group uses.**
- **If you have a BBS, please include the name, phone number, and the sysops name.**
- **Send us your newsletter, we enjoy reading them.**

Send to:
Amazing Users Unite
PiM Publications, Inc.
P.O. Box 869
Fall River, MA 02722-0869

Fred Fish Disk 209

Bowl This is Vern's entry for the 1988 Badge Killer Demo Contest. It is a Sculpt-Animate animation that shows three colored balls flying in circles above a mirrored bowl. Rendering the animation took about 2 weeks. Distributed in zoo format because of its size (zoo program included for easy unpacking). Author: Vern Staats

Dps A program designed to work with the PrintScript program, a commercial PostScript interpreter for the Amiga, to provide a page previewer. This is version 1.1 and includes source. Author: Allen Norskog

Fred Fish Disk 210

Calc A very nicely done scientific/programmer/plotter calculator. The scientific portion has most of the operations found on the more popular handhelds. The programmer portion has all the special hex/binary/decimal conversions as well as register operations like ASL, ROL, LSL, AND, OR, XOR, etc. The plotter portion will plot equations. Other features include 26 memories, full mouse or keyboard operation, pull-down menus, and iconization. Version 3.0, binary only. Author: Jimmy Yang

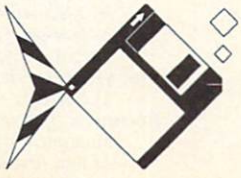
LabelPrint A program that allows you to easily print labels for your disks. Version 1.9, shareware, binary only (source available from author). Author: Andreas Krebs

NuHand An animation of a hand with fingernails scrapeing on a desktop, including sound effects. This is Bryan's entry for the 1988 Badge Killer Demo Contest. Binary only Author: Bryan Carey Gallivan

To Be Continued.....

In Conclusion

To the best of our knowledge, the materials in this library are freely distributable. This means they were either publicly posted and placed in the public domain by their authors, or they have restrictions published in their files to which we have adhered. If you become aware of any violation of the authors' wishes, please contact us by mail.



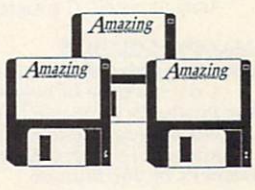
IMPORTANT NOTICE!

This list is compiled and published as a service to the Commodore Amiga community for informational purposes only. Its use is restricted to non-commercial groups only! Any duplication for commercial purposes is strictly forbidden. As a part of Amazing Computing™, this list is inherently copyrighted. Any infringement on this proprietary copyright without expressed written permission of the publishers will incur the full force of legal actions.

Any non-commercial Amiga user group wishing to duplicate this list should contact:

PiM Publications, Inc.
P.O.Box 869
Fall River, MA 02722

PiM Publications Inc. is extremely interested in helping any Amiga user groups in non-commercial support for the Amiga.



You Haven't Seen AC? Look What You Missed!

February 1989 Volume 4.2

AMAZING FEATURES

MovieSetter: The Next Generation
by Steve Gillmor
A new level in Amiga animation.

AmiForum *by Steve Pietrowicz*
The Amiga pays a mid-winter visit to Mickey's hometown for the first regional Amiga show.

Max Morehead Interview *by Richard Rae*
Rick talks to the creator of Moviesetter.

A Common User Interface for the Amiga
by Jim Bayless
Does the Amiga need a kinder, gentler interface? Vote now!

AMAZING REVIEWS

Superbase Professional *by Marion Deland*
A user's look at Superbase Pro.

Microfiche Filer Plus *by Ronald Courrier*
A multi-tasking database that uses ARexx to work with other programs

Torch 2081 *by Jeffery Scott Hall*
Fast action, and smooth graphics

AMAZING PROGRAMMING

SPY *by Steve Faiwiszewski*
Programming intrigue in Modula -2

Sync Tips*by Oren Sands*
Getting inside the genlock.

On the Crafting of Programs *by D J. Hankins*
Do we need a common standard for C programming? Look ANSI.

C Notes from the C Group *by Steven Kemp*
An introduction to unions

The Command Line *by Rich Falconburg*
ED, your Workbench Screen Editor

An Introduction to ARexx programming
by Steve Faiwizewski
Climbing the Towers of Hanoi.

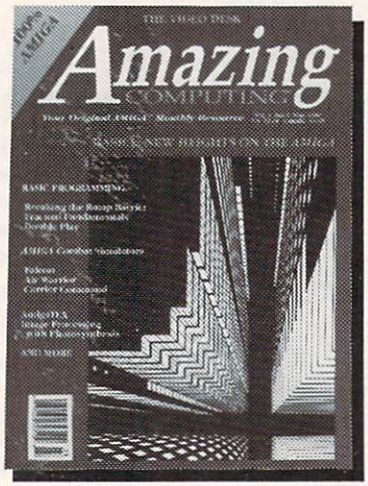
Crunchy Frog *by Jim Fiore*
Amiga-specific C programming.

AMAZING COLUMNS

Super Bug Bytes *by John Steiner*
A double dose of bug repellant.

New Products *by Michael Creeden*
What's New? Deluxe Print II, The Talking Animator, 3 digit fun, and more.

Roomers *by The Bandito*
Big resignation at Commodore, big budget cuts at a major, and the Bandito's fearless predictions.



March 1989 Volume 4.3

AMAZING FEATURES

Falcon *reviewed by Joe DiCara*
Latest from Spectrum Holobyte's hangar

Air Warrior *reviewed by Michael Mantino*
Hook up via modem and battle it out with international opponents!

Carrier Command
reviwed by Lawrence Lichtmann
Admiral Lichtman signs up for carrier duty

World of Commodore Toronto
by Ed Berkovitz
Ed files his report from Toronto

Fractal Fundamentals *by Paul Castonguay*
Experiment on the edge of a new science by creating your own fractals

AMAZING REVIEWS

Image Processing with Photosynthesis
by Gerald Hull
An experiment with a repertoire of AI image-processing techniques

Gizmoz 2.0 *by Steve Carter*
Steve Carter reviews version 2.0 of Digital Creations' box o' fun

AmigaTEX *by Barney Schwartz*
A page description language from Radical EyE software

flickerFixer *by Steve Bender*
Steve gives us a clear perspective on MicroWay's flickerFixer

AMAZING PROGRAMMING

Benchmark 1: Fully utilizing the MC68881
by Read Predmore
Part I: Turbocharging the savage benchmark

Breaking the Bmap Barrier *by Robert D'Asto*
Streamline your AmigaBASIC library access with Quick_Lib

Double Play *by Robert D'Asto*
AmigaBASIC program yeilds double vision.

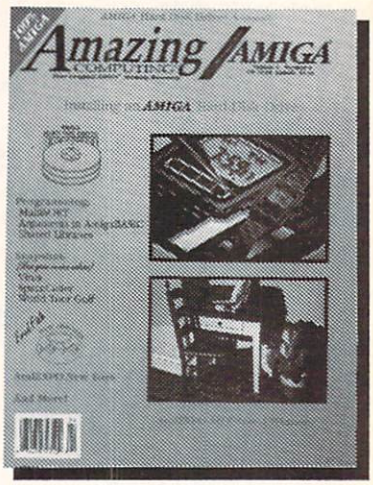
C Notes From the C Group *by Stephen Kemp*
A walk through preprocesor control lines

AMAZING COLUMNS

New Products...and other Neat Stuff
by Michael Creeden
3-M-dous! The latest from MichTron, MicroEd and Mindscape

The Video Desk *by Larry White*
The Amiga meets Nikon Camera

Roomers *by The Bandito*
Magneto-optical disks on the Horizon?... Amiga first stop for WordPerfect 6.0?...and the HAM paint wars continue.



April 1989 Volume 4.4

AMAZING FEATURES

AmiEXPO NY '89 *by Steve Gillmor*
Amiga takes a bite out of the Big Apple.

AmiEXPO Art and Video Contest Winners
by Steve Jacobs
Mouse & monitor replaced brush & canvas in this first-ever AmiEXPO art event.

Adding the Not-So-Hard Disk *by J P. Twardy*
Short on memory? Installing a Hard Drive was never so easy—or so cheap.

Hard Drives—an Introduction
by Jon A. Boulle
A straightforward, "no-Boulle" comparison of Hard Drives.

AMAZING REVIEWS

Tax Break *by Kim Schaffer*
OXXI's new tax program.

The Max Hard Drive Kit *by Donald W. Morgan*
Another inexpensive Hard Drive installation project, this time using Palomax's Max kit.

Menace *reviewed by Jeffrey Scott Hall*
Blast aliens & destroy the evil Draconia.

AMAZING PROGRAMMING

Sync Tips *by Oran J. Sands*
Oran presents a clearer picture regarding video and computer resolutions.

Passing Arguments *by Brian Zupke*
AmigaBASIC subprogram explains step-by-step how to pass data from the CLI to AmigaBASIC.

C Notes from the C Group *by Stephen Kemp*
A humorous view of the wacky world of programmers.

Creating a Shared Library *by John Baez*
Program for increased Amiga productivity.

MultiSort *by Steve Faiwiszewski*
Put your data in its place with this ultra organizing program.

AMAZING COLUMNS

New Products and Other Neat Stuff
by Michael Creeden
Face-off with Gretsky, play Picasso with Basic Art Encoder, balance your Desktop Budget, plus more.

Snapshot *by R. Brad Andrews*
Four exciting Amiga games are reviewed

Roomers *by The Bandito*
Amiga developers go for the bucks, Commodore stock up, & Paint Wars.

Bug Bytes *by John Steiner*
Soft-Logik's PageStream, and more.

Amazing COMPUTING Reader Service Card

Want to know more about Amazing Computing Advertisers and their products? Here's your chance! Complete the card below and drop it in the mail (U. S. only). We will process your request and forward it to the appropriate advertiser.

Quick, easy, and efficient—just like all the services of Amazing Computing: Your Original ***AMIGA*** Monthly Resource.

Amazing COMPUTING Reader Service Card

AC June '89 Valid Until 8/30/89
see page 96 for reference numbers

Name _____
Street _____
City _____ ST. _____ ZiP _____
Country _____

A. Do you own an Amiga?
O 1. Amiga 1000 O 4. Soon
O 2. Amiga 500 O 5. Not Yet
O 3. Amiga 2000 O 6. Just Looking

B. What Amiga hardware product do you plan to buy next?
O 1. Amiga 500 O 6. Printer
O 2. Amiga 2000 O 7. Modem
O 3. Memory Expansion O 8. Music Tool
O 4. Hard Drive O 9. Video Product
O 5. IBM Emulators (Sidecar or Bridgeboard) O 10. Other

C. What Amiga software product do you plan to buy next?
O 1. 'C' Language O 8. Spreadsheet
O 2. Forth Language O 9. Database
O 3. Modula-2 Language O 10. Financial
O 4. Assembly Language O 11. Video
O 5. BASIC Language O 12. Graphics
O 6. Entertainment O 13. Music
O 7. Telecommunications O 14. Other

D. Where do you buy your Amiga products?
O 1. Local Amiga Dealer O 3. Manufacturer
O 2. Discount Department Store O 4. Mail Order

E. Which type of articles do you like to see in Amazing Computing?
O 1. 'C' Language O 10. Programming How To's
O 2. Forth Language O 11. Business How To's
O 3. Modula-2 Language O 12. Video Articles
O 4. Assembly Language O 13. Graphics Articles
O 5. BASIC Language O 14. Music Articles
O 6. Game Reviews O 15. Hardware How To's
O 7. Business Reviews O 16. PDS Updates
O 8. Hardware Product Reviews O 17. Interviews
O 9. Software Product Reviews O 18. Other

F. Which articles would you like to see more of in Amazing Computing?
O 1. 'C' Language O 10. Programming How To's
O 2. Forth Language O 11. Business How To's
O 3. Modula-2 Language O 12. Video Articles
O 4. Assembly Language O 13. Graphics Articles
O 5. BASIC Language O 14. Music Articles
O 6. Game Reviews O 15. Hardware How To's
O 7. Business Reviews O 16. PDS Updates
O 8. Hardware Product Reviews O 17. Interviews
O 9. Software Product Reviews O 18. Other

G. Are you a subscriber to Amazing Computing?
O 1. Yes O 2. No

101

101	102	103	104	105	221	222	223	224	225
106	107	108	109	110	226	227	228	229	230
111	112	113	114	115	231	232	233	234	235
116	117	118	119	120	236	237	238	239	240
121	122	123	124	125	241	242	243	244	245
126	127	128	129	130	246	247	248	249	250
131	132	133	134	135	251	252	253	254	255
136	137	138	139	140	256	257	258	259	260
141	142	143	144	145	261	262	263	264	265
146	147	148	149	150	266	267	268	269	270
151	152	153	154	155	271	272	273	274	275
156	157	158	159	160	276	277	278	279	280
161	162	163	164	165	281	282	283	284	285
166	167	168	169	170	286	287	288	289	290
171	172	173	174	175	291	292	293	294	295
176	177	178	179	180	296	297	298	299	300
181	182	183	184	185	301	302	303	304	305
186	187	188	189	190	306	307	308	309	310
191	192	193	194	195	311	312	313	314	315
196	197	198	199	200	316	317	318	319	320
201	202	203	204	205	321	322	323	324	325
206	207	208	209	210	326	327	328	329	330
211	212	213	214	215	331	332	333	334	335
216	217	218	219	220	336	337	338	339	340

Amazing COMPUTING Reader Service Card

AC June '89 Valid Until 8/30/89
see page 96 for reference numbers

Name _____
Street _____
City _____ ST. _____ ZiP _____
Country _____

A. Do you own an Amiga?
O 1. Amiga 1000 O 4. Soon
O 2. Amiga 500 O 5. Not Yet
O 3. Amiga 2000 O 6. Just Looking

B. What Amiga hardware product do you plan to buy next?
O 1. Amiga 500 O 6. Printer
O 2. Amiga 2000 O 7. Modem
O 3. Memory Expansion O 8. Music Tool
O 4. Hard Drive O 9. Video Product
O 5. IBM Emulators (Sidecar or Bridgeboard) O 10. Other

C. What Amiga software product do you plan to buy next?
O 1. 'C' Language O 8. Spreadsheet
O 2. Forth Language O 9. Database
O 3. Modula-2 Language O 10. Financial
O 4. Assembly Language O 11. Video
O 5. BASIC Language O 12. Graphics
O 6. Entertainment O 13. Music
O 7. Telecommunications O 14. Other

D. Where do you buy your Amiga products?
O 1. Local Amiga Dealer O 3. Manufacturer
O 2. Discount Department Store O 4. Mail Order

E. Which type of articles do you like to see in Amazing Computing?
O 1. 'C' Language O 10. Programming How To's
O 2. Forth Language O 11. Business How To's
O 3. Modula-2 Language O 12. Video Articles
O 4. Assembly Language O 13. Graphics Articles
O 5. BASIC Language O 14. Music Articles
O 6. Game Reviews O 15. Hardware How To's
O 7. Business Reviews O 16. PDS Updates
O 8. Hardware Product Reviews O 17. Interviews
O 9. Software Product Reviews O 18. Other

F. Which articles would you like to see more of in Amazing Computing?
O 1. 'C' Language O 10. Programming How To's
O 2. Forth Language O 11. Business How To's
O 3. Modula-2 Language O 12. Video Articles
O 4. Assembly Language O 13. Graphics Articles
O 5. BASIC Language O 14. Music Articles
O 6. Game Reviews O 15. Hardware How To's
O 7. Business Reviews O 16. PDS Updates
O 8. Hardware Product Reviews O 17. Interviews
O 9. Software Product Reviews O 18. Other

G. Are you a subscriber to Amazing Computing?
O 1. Yes O 2. No

102

101	102	103	104	105	221	222	223	224	225
106	107	108	109	110	226	227	228	229	230
111	112	113	114	115	231	232	233	234	235
116	117	118	119	120	236	237	238	239	240
121	122	123	124	125	241	242	243	244	245
126	127	128	129	130	246	247	248	249	250
131	132	133	134	135	251	252	253	254	255
136	137	138	139	140	256	257	258	259	260
141	142	143	144	145	261	262	263	264	265
146	147	148	149	150	266	267	268	269	270
151	152	153	154	155	271	272	273	274	275
156	157	158	159	160	276	277	278	279	280
161	162	163	164	165	281	282	283	284	285
166	167	168	169	170	286	287	288	289	290
171	172	173	174	175	291	292	293	294	295
176	177	178	179	180	296	297	298	299	300
181	182	183	184	185	301	302	303	304	305
186	187	188	189	190	306	307	308	309	310
191	192	193	194	195	311	312	313	314	315
196	197	198	199	200	316	317	318	319	320
201	202	203	204	205	321	322	323	324	325
206	207	208	209	210	326	327	328	329	330
211	212	213	214	215	331	332	333	334	335
216	217	218	219	220	336	337	338	339	340

Amazing COMPUTING Reader Service Card

Want to know more about Amazing Computing Advertisers and their products? Here's your chance! Complete the card below and drop it in the mail (U. S. only). We will process your request and forward it to the appropriate advertiser.

Quick, easy, and efficient—just like all the services of Amazing Computing: Your Original ***AMIGA*** Monthly Resource.

Amazing COMPUTING Reader Service Card

AC June '89 Valid Until 8/30/89
see page 96 for reference numbers

Name _____
Street _____
City _____ ST. _____ ZiP _____
Country _____

A. Do you own an Amiga?
- O 1. Amiga 1000
- O 2. Amiga 500
- O 3. Amiga 2000
- O 4. Soon
- O 5. Not Yet
- O 6. Just Looking

B. What Amiga hardware product do you plan to buy next?
- O 1. Amiga 500
- O 2. Amiga 2000
- O 3. Memory Expansion
- O 4. Hard Drive
- O 5. IBM Emulators (Sidecar or Bridgeboard)
- O 6. Printer
- O 7. Modem
- O 8. Music Tool
- O 9. Video Product
- O 10. Other

C. What Amiga software product do you plan to buy next?
- O 1. 'C' Language
- O 2. Forth Language
- O 3. Modula-2 Language
- O 4. Assembly Language
- O 5. BASIC Language
- O 6. Entertainment
- O 7. Telecommunications
- O 8. Spreadsheet
- O 9. Database
- O 10. Financial
- O 11. Video
- O 12. Graphics
- O 13. Music
- O 14. Other

D. Where do you buy your Amiga products?
- O 1. Local Amiga Dealer
- O 2. Discount Department Store
- O 3. Manufacturer
- O 4. Mail Order

E. Which type of articles do you like to see in Amazing Computing?
- O 1. 'C' Language
- O 2. Forth Language
- O 3. Modula-2 Language
- O 4. Assembly Language
- O 5. BASIC Language
- O 6. Game Reviews
- O 7. Business Reviews
- O 8. Hardware Product Reviews
- O 9. Software Product Reviews
- O 10. Programming How To's
- O 11. Business How To's
- O 12. Video Articles
- O 13. Graphics Articles
- O 14. Music Articles
- O 15. Hardware How To's
- O 16. PDS Updates
- O 17. Interviews
- O 18. Other

F. Which articles would you like to see more of in Amazing Computing?
- O 1. 'C' Language
- O 2. Forth Language
- O 3. Modula-2 Language
- O 4. Assembly Language
- O 5. BASIC Language
- O 6. Game Reviews
- O 7. Business Reviews
- O 8. Hardware Product Reviews
- O 9. Software Product Reviews
- O 10. Programming How To's
- O 11. Business How To's
- O 12. Video Articles
- O 13. Graphics Articles
- O 14. Music Articles
- O 15. Hardware How To's
- O 16. PDS Updates
- O 17. Interviews
- O 18. Other

G. Are you a subscriber to Amazing Computing?
- O 1. Yes
- O 2. No

101

101	102	103	104	105	221	222	223	224	225
106	107	108	109	110	226	227	228	229	230
111	112	113	114	115	231	232	233	234	235
116	117	118	119	120	236	237	238	239	240
121	122	123	124	125	241	242	243	244	245
126	127	128	129	130	246	247	248	249	250
131	132	133	134	135	251	252	253	254	255
136	137	138	139	140	256	257	258	259	260
141	142	143	144	145	261	262	263	264	265
146	147	148	149	150	266	267	268	269	270
151	152	153	154	155	271	272	273	274	275
156	157	158	159	160	276	277	278	279	280
161	162	163	164	165	281	282	283	284	285
166	167	168	169	170	286	287	288	289	290
171	172	173	174	175	291	292	293	294	295
176	177	178	179	180	296	297	298	299	300
181	182	183	184	185	301	302	303	304	305
186	187	188	189	190	306	307	308	309	310
191	192	193	194	195	311	312	313	314	315
196	197	198	199	200	316	317	318	319	320
201	202	203	204	205	321	322	323	324	325
206	207	208	209	210	326	327	328	329	330
211	212	213	214	215	331	332	333	334	335
216	217	218	219	220	336	337	338	339	340

Amazing COMPUTING Reader Service Card

AC June '89 Valid Until 8/30/89
see page 96 for reference numbers

Name _____
Street _____
City _____ ST. _____ ZiP _____
Country _____

A. Do you own an Amiga?
- O 1. Amiga 1000
- O 2. Amiga 500
- O 3. Amiga 2000
- O 4. Soon
- O 5. Not Yet
- O 6. Just Looking

B. What Amiga hardware product do you plan to buy next?
- O 1. Amiga 500
- O 2. Amiga 2000
- O 3. Memory Expansion
- O 4. Hard Drive
- O 5. IBM Emulators (Sidecar or Bridgeboard)
- O 6. Printer
- O 7. Modem
- O 8. Music Tool
- O 9. Video Product
- O 10. Other

C. What Amiga software product do you plan to buy next?
- O 1. 'C' Language
- O 2. Forth Language
- O 3. Modula-2 Language
- O 4. Assembly Language
- O 5. BASIC Language
- O 6. Entertainment
- O 7. Telecommunications
- O 8. Spreadsheet
- O 9. Database
- O 10. Financial
- O 11. Video
- O 12. Graphics
- O 13. Music
- O 14. Other

D. Where do you buy your Amiga products?
- O 1. Local Amiga Dealer
- O 2. Discount Department Store
- O 3. Manufacturer
- O 4. Mail Order

E. Which type of articles do you like to see in Amazing Computing?
- O 1. 'C' Language
- O 2. Forth Language
- O 3. Modula-2 Language
- O 4. Assembly Language
- O 5. BASIC Language
- O 6. Game Reviews
- O 7. Business Reviews
- O 8. Hardware Product Reviews
- O 9. Software Product Reviews
- O 10. Programming How To's
- O 11. Business How To's
- O 12. Video Articles
- O 13. Graphics Articles
- O 14. Music Articles
- O 15. Hardware How To's
- O 16. PDS Updates
- O 17. Interviews
- O 18. Other

F. Which articles would you like to see more of in Amazing Computing?
- O 1. 'C' Language
- O 2. Forth Language
- O 3. Modula-2 Language
- O 4. Assembly Language
- O 5. BASIC Language
- O 6. Game Reviews
- O 7. Business Reviews
- O 8. Hardware Product Reviews
- O 9. Software Product Reviews
- O 10. Programming How To's
- O 11. Business How To's
- O 12. Video Articles
- O 13. Graphics Articles
- O 14. Music Articles
- O 15. Hardware How To's
- O 16. PDS Updates
- O 17. Interviews
- O 18. Other

G. Are you a subscriber to Amazing Computing?
- O 1. Yes
- O 2. No

102

101	102	103	104	105	221	222	223	224	225
106	107	108	109	110	226	227	228	229	230
111	112	113	114	115	231	232	233	234	235
116	117	118	119	120	236	237	238	239	240
121	122	123	124	125	241	242	243	244	245
126	127	128	129	130	246	247	248	249	250
131	132	133	134	135	251	252	253	254	255
136	137	138	139	140	256	257	258	259	260
141	142	143	144	145	261	262	263	264	265
146	147	148	149	150	266	267	268	269	270
151	152	153	154	155	271	272	273	274	275
156	157	158	159	160	276	277	278	279	280
161	162	163	164	165	281	282	283	284	285
166	167	168	169	170	286	287	288	289	290
171	172	173	174	175	291	292	293	294	295
176	177	178	179	180	296	297	298	299	300
181	182	183	184	185	301	302	303	304	305
186	187	188	189	190	306	307	308	309	310
191	192	193	194	195	311	312	313	314	315
196	197	198	199	200	316	317	318	319	320
201	202	203	204	205	321	322	323	324	325
206	207	208	209	210	326	327	328	329	330
211	212	213	214	215	331	332	333	334	335
216	217	218	219	220	336	337	338	339	340

Subscribe and More!

Amazing is now available by Visa and MasterCard
dial 1-800-345-3360
or use the form below.

Subscribe TODAY and save over 49% on Amazing Computing!

Please use this order form when subscribing to Amazing Computing™, ordering Back Issues, or ordering Amiga™ Public Domain Software

Name_____
Street_____
City_____ St._____ Zip_____
*Charge my ☐ Visa ☐ MC #_____ Expiration Date _____
Signature _____ All Charges are subject to a $20.00 minimum.

PROPER ADDRESS REQUIRED. In order to expedite and guarantee your order, all large Public Domain Software orders, as well as most Back issue orders, are shipped by United Parcel Service. UPS requires that all packages be addressed to a street address for correct delivery.

Subscription: $_____

Please circle the appropriate item: **New Subscription** **Renewal**

Please start my subscription to Amazing Computing™ with the next available issue or renew my current subscription. I have enclosed $24.00 for 12 issues in the U.S. ($36.00 Canada and Mexico, $44.00 foreign surface). All funds must be in U.S. Currency on a U.S. Bank

Back Issues:
$5.00 each US, $6.00 each Canada and Mexico, $7.00 each Foreign Surface.

Please circle your Back Issue choices below:

Vol1.1	Vol1.2	Vol1.3	Vol1.4	Vol1.5	Vol1.6	Vol1.7	Vol1.8	Vol1.9	Vol2.1	Vol2.2
Vol2.3	Vol2.4	Vol2.5	Vol2.6	Vol2.7	Vol2.8	Vol2.9	Vol2.10	Vol2.11	Vol2.12	Vol3.1
Vol3.2	Vol3.3	Vol3.4	Vol3.5	Vol3.6	Vol3.7	Vol3.8	Vol3.9	Vol3.10	Vol3.11	Vol3.12
Vol4.1	Vol4.2	Vol4.3	Vol4.4	Vol4.5	(Limited Offer) AC Product Guide Spring '89					

Back Issues: $_____

Public Domain Software:
$6.00 each for subscribers (yes, even the new ones!)
$7.00 each for non subscribers
(three disk minimum on all foreign orders)

Please circle your Public Domain Software choices below:

Amazing on Disk: A#1. . .Source & Listings V3.8& V3.9 A#2. . .Source & Listings V4.4
A#3. . .Source & Listings V4.5 & V4.6 **InNOCKulation Disk:** IN#1...Virus protection

Amicus:

A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17
A18	A19	A20	A21	A22	A23	A24	A25	A26								

Fred Fish:

FF1	FF2	FF3	FF4	FF5	FF6	FF7	FF8	FF9	FF10	FF11	FF12	FF13	FF14	FF15	FF16	FF17
FF18	FF19	FF20	FF21	FF22	FF23	FF24	FF25	FF26	FF27	FF28	FF29	FF30	FF31	FF32	FF33	FF34
FF35	FF36	FF37	FF38	FF39	FF40	FF41	FF42	FF43	FF44	FF45	FF46	FF47	FF48	FF49	FF50	FF51
FF52	FF53	FF54	FF55	FF56	FFNA	FF58	FF59	FF60	FF61	FF62	FF63	FF64	FF65	FF66	FF67	FF68
FF69	FF70	FF71	FF72	FF73	FF74	FF75	FF76	FF77	FF78	FF79	FFNA	FF81	FF82	FF83	FF84	FF85
FF86	FF87	FFNA	FF89	FF90	FF91	FF92	FF93	FF94	FF95	FF96	FF97	FF98	FF99	FF100	FF101	FF102
FF103	FF104	FF105	FF106	FF107	FF108	FF109	FF110	FF111	FF112	FF113	FF114	FF115	FF116	FF117	FF118	FF119
FF120	FF121	FF122	FF123	FF124	FF125	FF126	FF127	FF128	FF129	FF130	FF131	FF132	FF133	FF134	FF135	FF136
FF137	FF138	FF139	FF140	FF141	FF142	FF143	FF144	FF145	FF146	FF147	FF148	FF149	FF150	FF151	FF152	FF153
FF154	FF155	FF156	FF157	FF158	FF159	FF160	FF161	FF162	FF163	FF164	FF165	FF166	FF167	FF168	FF169	FF170
FF171	FF172	FF173	FF174	FF175	FF176	FF177	FF178	FF179	FF180	FF181	FF182	FF183	FF184	FF185	FF186	FF187
FF188	FF189	FF190	FF191	FF192	FF193	FF194	FF195	FF196	FF197	FF198	FF199	FF200	FF201	FF202	FF203	FF204
FF205	FF206	FF207	FF208	FF209	FF210	(NA Denotes disks removed from the collection)										

PDS Disks: $_____

Please complete this form and mail with check or money order to:

PiM Publications, Inc.
P.O. Box 869
Fall River, MA 02722-0869

Total: $_____

Please allow 4 to 6 weeks for delivery

Subscribe and More!

Amazing is now available by Visa and MasterCard
dial 1-800-345-3360
or use the form below.

Subscribe TODAY and save over 49% on Amazing Computing!

Please use this order form when subscribing to Amazing Computing™, ordering Back Issues, or ordering Amiga™ Public Domain Software

Name_____
Street_____
City_____ St._____ Zip_____
*Charge my ☐ Visa ☐ MC #_____ Expiration Date _____
Signature _____ All Charges are subject to a $20.00 minimum.

PROPER ADDRESS REQUIRED. In order to expedite and guarantee your order, all large Public Domain Software orders, as well as most Back issue orders, are shipped by United Parcel Service. UPS requires that all packages be addressed to a street address for correct delivery.

Subscription: $_____

Please circle the appropriate item: **New Subscription** **Renewal**

Please start my subscription to Amazing Computing™ with the next available issue or renew my current subscription. I have enclosed $24.00 for 12 issues in the U.S. ($36.00 Canada and Mexico, $44.00 foreign surface). All funds must be in U.S. Currency on a U.S. Bank

Back Issues:
$5.00 each US, $6.00 each Canada and Mexico, $7.00 each Foreign Surface.

Please circle your Back Issue choices below:

Vol1.1	Vol1.2	Vol1.3	Vol1.4	Vol1.5	Vol1.6	Vol1.7	Vol1.8	Vol1.9	Vol2.1	Vol2.2
Vol2.3	Vol2.4	Vol2.5	Vol2.6	Vol2.7	Vol2.8	Vol2.9	Vol2.10	Vol2.11	Vol2.12	Vol3.1
Vol3.2	Vol3.3	Vol3.4	Vol3.5	Vol3.6	Vol3.7	Vol3.8	Vol3.9	Vol3.10	Vol3.11	Vol3.12
Vol4.1	Vol4.2	Vol4.3	Vol4.4	Vol4.5	(Limited Offer) AC Product Guide Spring '89					

Back Issues: $_____

Public Domain Software:
$6.00 each for subscribers (yes, even the new ones!)
$7.00 each for non subscribers
(three disk minimum on all foreign orders)

Please circle your Public Domain Software choices below:

Amazing on Disk: A#1. . .Source & Listings V3.8& V3.9 A#2. . .Source & Listings V4.4
A#3. . .Source & Listings V4.5 & V4.6 **InNOCKulation Disk:** IN#1...Virus protection

Amicus:

A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17
A18	A19	A20	A21	A22	A23	A24	A25	A26								

Fred Fish:

FF1	FF2	FF3	FF4	FF5	FF6	FF7	FF8	FF9	FF10	FF11	FF12	FF13	FF14	FF15	FF16	FF17
FF18	FF19	FF20	FF21	FF22	FF23	FF24	FF25	FF26	FF27	FF28	FF29	FF30	FF31	FF32	FF33	FF34
FF35	FF36	FF37	FF38	FF39	FF40	FF41	FF42	FF43	FF44	FF45	FF46	FF47	FF48	FF49	FF50	FF51
FF52	FF53	FF54	FF55	FF56	FFNA	FF58	FF59	FF60	FF61	FF62	FF63	FF64	FF65	FF66	FF67	FF68
FF69	FF70	FF71	FF72	FF73	FF74	FF75	FF76	FF77	FF78	FF79	FFNA	FF81	FF82	FF83	FF84	FF85
FF86	FF87	FFNA	FF89	FF90	FF91	FF92	FF93	FF94	FF95	FF96	FF97	FF98	FF99	FF100	FF101	FF102
FF103	FF104	FF105	FF106	FF107	FF108	FF109	FF110	FF111	FF112	FF113	FF114	FF115	FF116	FF117	FF118	FF119
FF120	FF121	FF122	FF123	FF124	FF125	FF126	FF127	FF128	FF129	FF130	FF131	FF132	FF133	FF134	FF135	FF136
FF137	FF138	FF139	FF140	FF141	FF142	FF143	FF144	FF145	FF146	FF147	FF148	FF149	FF150	FF151	FF152	FF153
FF154	FF155	FF156	FF157	FF158	FF159	FF160	FF161	FF162	FF163	FF164	FF165	FF166	FF167	FF168	FF169	FF170
FF171	FF172	FF173	FF174	FF175	FF176	FF177	FF178	FF179	FF180	FF181	FF182	FF183	FF184	FF185	FF186	FF187
FF188	FF189	FF190	FF191	FF192	FF193	FF194	FF195	FF196	FF197	FF198	FF199	FF200	FF201	FF202	FF203	FF204
FF205	FF206	FF207	FF208	FF209	FF210	(NA Denotes disks removed from the collection)										

PDS Disks: $_____

Please complete this form and mail with check or money order to:

PiM Publications, Inc.
P.O. Box 869
Fall River, MA 02722-0869

Total: $_____

Please allow 4 to 6 weeks for delivery

Amaze Me

Please use this order form when subscribing to Amazing Computing™, ordering Back Issues, or ordering Amiga™ Public Domain Software

Name_____
Street_____
City_____ St._____ Zip_____
Charge my ☐Visa ☐MC #_____ Expiration Date _____
Signature _____ All Charges are subject to a $20.00 minimum

PROPER ADDRESS REQUIRED. In order to expedite and guarantee your order, all large Public Domain Software orders, as well as most Back issue orders, are shipped by United Parcel Service. UPS requires that all packages be addressed to a street address for correct delivery.

Subscription: $_____

Please circle the appropriate item: **New Subscription** **Renewal**

Please start my subscription to Amazing Computing™ with the next available issue or renew my current subscription. I have enclosed $24.00 for 12 issues in the U.S. ($36.00 Canada and Mexico, $44.00 foreign surface). All funds must be in U.S. Currency on a U.S. Bank

Back Issues:
$5.00 each US, $6.00 each Canada and Mexico, $7.00 each Foreign Surface.

Please circle your Back Issue choices below:

Vol1.1	Vol1.2	Vol1.3	Vol1.4	Vol1.5	Vol1.6	Vol1.7	Vol1.8	Vol1.9	Vol2.1	Vol2.2
Vol2.3	Vol2.4	Vol2.5	Vol2.6	Vol2.7	Vol2.8	Vol2.9	Vol2.10	Vol2.11	Vol2.12	Vol3.1
Vol3.2	Vol3.3	Vol3.4	Vol3.5	Vol3.6	Vol3.7	Vol3.8	Vol3.9	Vol3.10	Vol3.11	Vol3.12
Vol4.1	Vol4.2	Vol4.3	Vol4.4	Vol4.5	(Limited Offer) AC Product Guide Spring '89					

Back Issues: $_____

Public Domain Software:
$6.00 each for subscribers (yes, even the new ones!)
$7.00 each for non subscribers
(three disk minimum on all foreign orders)

Please circle your Public Domain Software choices below:

Amazing on Disk: A#1. . .Source & Listings V3.8& V3.9
A#2. . .Source & Listings V4.4
A#3. . .Source & Listings V4.5 & V4.6
InNOCKulation Disk: IN#1...Virus protection

Amicus:

A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17
A18	A19	A20	A21	A22	A23	A24	A25	A26								

Fred Fish:

FF1	FF2	FF3	FF4	FF5	FF6	FF7	FF8	FF9	FF10	FF11	FF12	FF13	FF14	FF15	FF16	FF17
FF18	FF19	FF20	FF21	FF22	FF23	FF24	FF25	FF26	FF27	FF28	FF29	FF30	FF31	FF32	FF33	FF34
FF35	FF36	FF37	FF38	FF39	FF40	FF41	FF42	FF43	FF44	FF45	FF46	FF47	FF48	FF49	FF50	FF51
FF52	FF53	FF54	FF55	FF56	FFNA	FF58	FF59	FF60	FF61	FF62	FF63	FF64	FF65	FF66	FF67	FF68
FF69	FF70	FF71	FF72	FF73	FF74	FF75	FF76	FF77	FF78	FF79	FFNA	FF81	FF82	FF83	FF84	FF85
FF86	FF87	FFNA	FF89	FF90	FF91	FF92	FF93	FF94	FF95	FF96	FF97	FF98	FF99	FF100	FF101	FF102
FF103	FF104	FF105	FF106	FF107	FF108	FF109	FF110	FF111	FF112	FF113	FF114	FF115	FF116	FF117	FF118	FF119
FF120	FF121	FF122	FF123	FF124	FF125	FF126	FF127	FF128	FF129	FF130	FF131	FF132	FF133	FF134	FF135	FF136
FF137	FF138	FF139	FF140	FF141	FF142	FF143	FF144	FF145	FF146	FF147	FF148	FF149	FF150	FF151	FF152	FF153
FF154	FF155	FF156	FF157	FF158	FF159	FF160	FF161	FF162	FF163	FF164	FF165	FF166	FF167	FF168	FF169	FF170
FF171	FF172	FF173	FF174	FF175	FF176	FF177	FF178	FF179	FF180	FF181	FF182	FF183	FF184	FF185	FF186	FF187
FF188	FF189	FF190	FF191	FF192	FF193	FF194	FF195	FF196	FF197	FF198	FF199	FF200	FF201	FF202	FF203	FF204
FF205	FF206	FF207	FF208	FF209	FF210	(NA Denotes disks removed from the collection)										

PDS Disks: $_____

Please complete this form and mail with check or money order to:

PiM Publications, Inc.
P.O. Box 869
Fall River, MA 02722-0869
1-800-345-3360

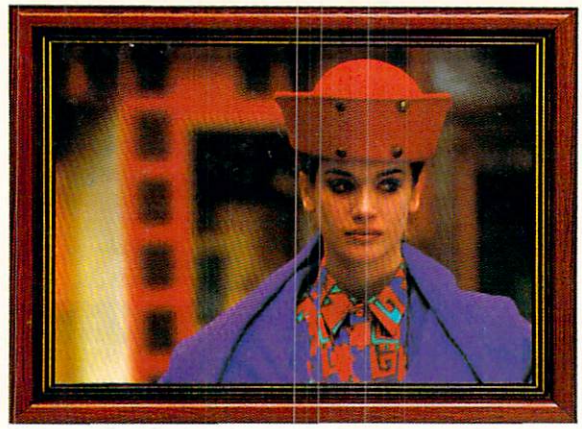
Total: $_____

Please allow 4 to 6 weeks for delivery

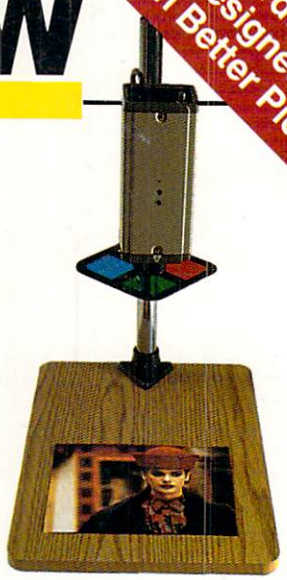
DIGI-VIEW GOLD

All New!
Hardware and Software
Designed for A500/2000—
Even Better Pictures Than Before

1.



2.



Actual 4096 color Digi-View Gold picture

3. Simply the Best.

The all new Digi-View Gold is the best video digitizer for the Amiga. Period. Nothing else even comes close. Why? The secret is that Digi-View Gold captures 2.1 million colors in memory, giving you an incredible 100,000 apparent colors on screen simultaneously.

And it's easy to use. Just focus your video camera on any object or picture, and in seconds Digi-View Gold turns it into Amiga graphics that glow with vibrant color and clarity. Whether you are creating graphics for desktop publishing, presentations, video, or just for fun, Digi-View Gold gives you dazzling images with amazing simplicity.

Digi-View Gold is designed specifically for the Amiga 500 and 2000, and plugs directly into the parallel port. Digi-View Gold's powerful image capture and manipulation software (version 3.0) now has complete control of color and sharpness, full overscan, extra halfbrite, and a special line art mode for desktop publishing.

Only Digi-View Gold:

- Can digitize in all Amiga resolution modes from 320x200 up to 768x480 (full hi-res overscan)
- Uses 2 to 4096 colors (including extra halfbrite)
- Uses exclusive Enhanced HAM for super fine detail
- Is 100% IFF compatible and works with any graphics software
- Can digitize 21 bits per pixel (2.1 million colors) for the highest quality images possible
- Has advanced dithering routines that give an apparent 100,000 colors on screen simultaneously
- Has powerful Image processing controls for complete IFF picture manipulation

If you want the highest quality graphics for your Amiga, as easy as 1, 2, 3; then you need the new version of the best selling video digitizer of all time: Digi-View Gold.

Only $199.95

Digi-View Gold is available now at your local Amiga dealer.
Or call 1-800-843-8934

NewTek INCORPORATED

*Requires standard gender changer for use with Amiga 1000. Video camera required; not included. NewTek sells a video camera, copy stand, and the Digi-Droid automated filter wheel for Digi-View Gold. If your local retailer doesn't carry these products, call us at 913-354-1146. Digi-View Gold is a trademark of NewTek, Inc. Amiga is a trademark of Commodore-Amiga, Inc. Be seeing you!!